Notes

Marco Codato

April 13, 2022

1 Introduction

Moved to the GitHub repository `codatomrc/tesi`.

2 Background spectrum extraction

2.1 Automatic detection

- Integrate over all the wavelenghts by summing the data in the dispersion direction. In this way a can synthetically see the flux along the spatial direction, regardless the source (i.e. continuum or line).
- Detect the peaks the integrated flux using the function `scipy.signal.find_peaks`. This function finds the local maxima of an 1D signal.

  On the current data it was convenient to filter the detected peak to have a prominence (in this case $\sim$ height from the background) higher than 0.5% the highest peak on the signal. This helped to neglect local maxima due to spatial noise fluctuations.
- Measure the width of each peak with `scipy.signal.peak_widths`. This function is designed to work with noisy signal and several maxima and minima, instead of just some maxima above a flat background. I set the option `rel_height=0.5` to measure the FWHM of each peack instead of the width at the base, which turned to be not very solid for the data we had.
- Mask the regions around the peaks. After some testing, the best strated (with the data available) is to remove a region of width $5\times$ FWHM around each peak.

  Note if peak profile fits a gaussian appriximation (i.e. sharp peaks and/or high SNR and/ord profile intrisically gaussian) then 5 FWHM correspond span to 5.89σ from the maximum, when the flux is dimmed by a factor 10^{-8}.

  Such an high span is required as many sources have not gaussian profiles and in particular might present bright wings wrt the gaussian prediction (e.g. galaxies have a bright nearly gaussian core plus faint but extended spiral arms).
- Extract the data of the background as those outside the region of the peaks. These are the areas that will be used for the analysis.

Note that due to the variety of luminosity profiles of astronomical sources it is not very efficient to develope a qualitative approach to compute the best width. This empirical treatment is good enough (if it will prove to be solid with more data though).

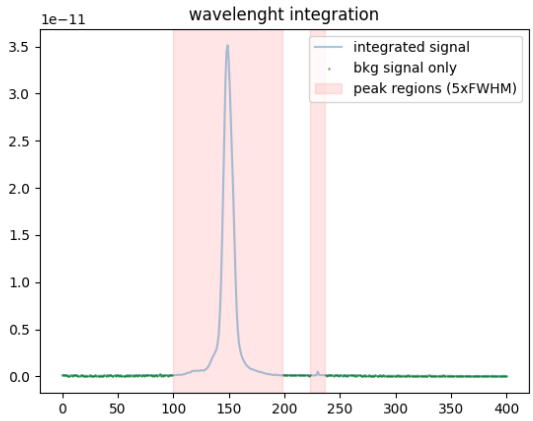
2.2 Preliminary results

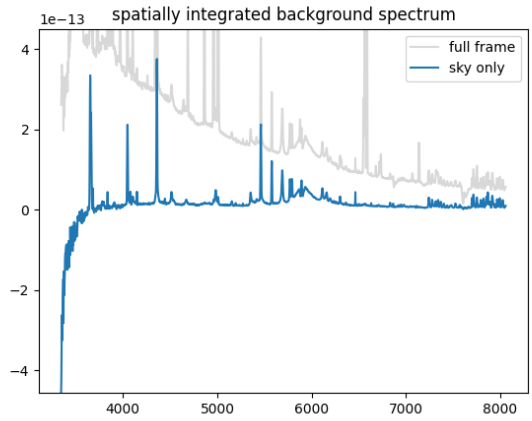
Here the results with the two frames available.

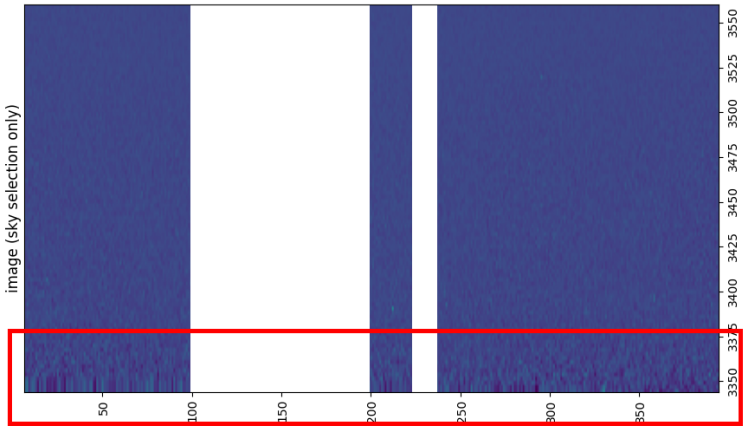
NGC2392 (2006/ima_010). The Eskimo Nebula. There is a sharp peak with strong wings. Probably another very faint source is present in the field. The script does recognize both the signals and removes them.

We integrate over all the position along the slit where we no astronomical sources/signals are present. Then we plot it against the wavelenght to obtain the spectrum of the background (atmospheric emission + light pollution).

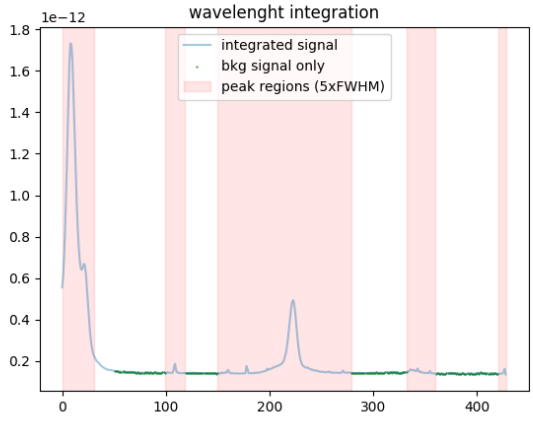
In the removal they comes out negative fluxes in the bluer hand of the spectrum. We realize that the wavelenght range in this frame begins at $\sim 3344\,\text{Å}$ which considerable as UV radiation. It is not so surprising to find inaccurate results at those extreme wavelenghts.

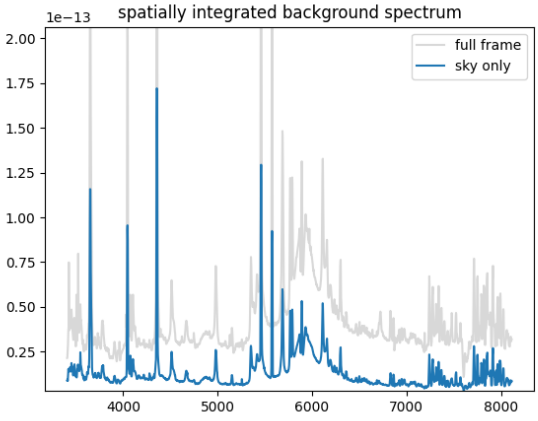






Ark564 (2006/ima_015). In this case the sources along the slit seems to be several but the script seems to be able to manage all of them. In this case the spectrum of the background do not seem to present any issue.





2.3 Comments on the first results

Wavelenght limitations. It would be convenient to set a wavelenght treshold for the analized data. Wavelenghts outside the spectrograph+telescope working range may led to biased measurements.

Frame limitations(?) I wonder wether the whole CCD area is suitable for collecting data or it would be better/necessary to neglect some specific regions, both in the spatial and dispersion directions.

What about the lines "`CCDSEC`" and "`BIASSEC`" in the header of the frames?

3 Appendix: Python source code

The code I am using.

```
import numpy as np
import matplotlib.pyplot as plt
from astropy.io import fits
import astropy.coordinates as coord
from astropy import units as u
from astroquery.simbad import Simbad
from scipy.signal import find_peaks, peak_widths

#open a FITS file
file = 'Asiago_nightsky/2006/ima_015.fc.fits'
hdul = fits.open(file)
hdr = hdul[0].header

#extract wavelenght information from the header
LAMBDA0 = hdr['CRVAL1']
DELTA = hdr['CDELT1']
NAXIS1 = hdr['NAXIS1']
LAMBDA = np.arange(LAMBDA0, LAMBDA0+NAXIS1*DELTA, DELTA)
NAXIS2 = hdr['NAXIS2']

##################
#integrate along the wavelenghts (dispersion direction)
integr = 0
for i in range(2047):
    integr = integr + hdul[0].data[:,i]

#find peaks and measure their widths
peaks,properties = find_peaks(integr, prominence=max(integr)/200.)
peak_FWHM = peak_widths(integr, peaks, rel_height=0.5)[0]

#set left and right boundaries of the source region along the slit
width_mult = 5
left_width = peaks-peak_FWHM*width_mult
right_width = peaks+peak_FWHM*width_mult

##########################
#remove overlapping ranges

#compress left and right boundaries into a single array
widths = np.array([left_width.T, right_width.T]).T
widths = np.reshape(widths, 2*len(left_width))

#find and mark overlaps
for i in range(len(widths)-1):
    if widths[i] > widths[i+1]:
        widths[i] = np.nan
        widths[i+1] = np.nan
#shrink the array to unmasked values
widths=widths[~np.isnan(widths)]

#reshape the array into the original two
left_width, right_width = np.reshape(widths, (int(len(widths)/2),2)).T

#remove values beyond the CCD size limits
if left_width[0] < 0:
    left_width[0] = 0
if right_width[-1] > NAXIS2:
    right_width[-1] = NAXIS2

########################
#mask the source/background regions
sign_sel = np.zeros(np.shape(integr), dtype=bool)
for i in range(len(integr)):
    for peak,width in zip(peaks,peak_FWHM):
        if abs(i-peak) < width*width_mult:
            sign_sel[i] = True
bkg_sel = ~sign_sel

#plot the integrated flux, show source and bkg regions
if 1 == True:
    plt.title('wavelenght integration')
    plt.plot(integr, alpha=0.4) #integrated flux
    plt.scatter(np.arange(len(integr))[bkg_sel],
```

```
                integr[bkg_sel],
                s=0.2, c='green') #select bkg

    for i in range(len(left_width)): #show all the source regions
        plt.axvspan(left_width[i],
                    right_width[i],
                    alpha=0.1, color='red')

    plt.legend(['integrated signal','bkg signal only', f'peak regions ({width_mult}xFWHM)'])
    plt.show()

###################
#integrated spectrum (along the slit)
spectrum = hdul[0].data #original data
total = np.sum(spectrum, axis = 0) #integration along the slit
sky = np.sum(spectrum[bkg_sel,:], axis = 0) #integration of bkg rows only

#plot the spatially integrated spectrum of the bkg
if 1 == True:
    plt.title('spatially integrated background spectrum')
    plt.plot(LAMBDA, total, label='full frame', color='gray', alpha=0.3)
    plt.plot(LAMBDA, sky, label='sky only')
    plt.ylim(min(sky),1.2*max(sky))
    plt.legend()
    plt.show()

###################
#extract only the bkg rows

ma_spectrum = spectrum #set masked data
for i,row in enumerate(bkg_sel):
    #cancel data from the source rows
    if row == 0:
        ma_spectrum[i,:] = np.nan

#plot as image the bkr rows only
if 1 == True:
    plt.title('image (sky selection only)')
    plt.imshow(ma_spectrum, extent = [LAMBDA[0], LAMBDA[-1], NAXIS2, 0])
    plt.show()

###################
#sava masked data in a new FITS file
if 1 == False:
    hdr.set('prova', 'prova')
    new_hdu = fits.PrimaryHDU(ma_spectrum)
    new_hdul = fits.HDUList([new_hdu])
    new_hdul[0].header = hdr

    file_new = file[:-5]+'.bkg.fits'

    new_hdul.writeto(file_new, overwrite=True)
    print('saved')
```