Notes

Marco Codato

April 26, 2022

1 Introduction

Moved to the GitHub repository `codatomrc/tesi`.

2 Background spectrum extraction

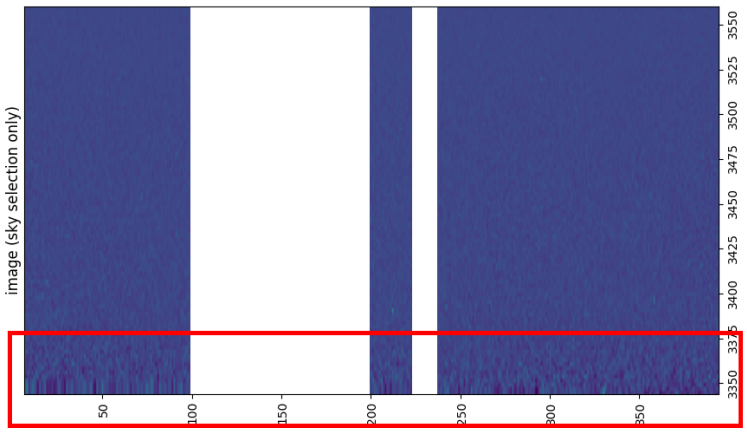
2.1 Automatic detection

The script I am writing works as follows.

- Automatically explores a directory and find all the raw files, marked as `filename.fc.fits`. All the files are then processed one by one.
- Far each frame relevant information is extracted from the header (hdr). In particular the script requires
  - information about the wavelength of each pixel (initial wavelength, increment of each pixel and width of the detector and horizontal binning factor),
  - slit width and length (thus the detector height an vertical binning factor, aside the CCD scale),
  - year of observation
- For each frame a preliminary integration over all the wavelenghts is performed by summing the data in the file along the dispersion direction. This is an efficient way of detecting astronomical sources, that are spatially limited, against the background, supposed uniform along the slit.
- I preliminarly decided to cut blue wavelengths at 3500 Å. This value was bull-eyed looking at bkg spectra and relizing that this is the treshold where the noise becomes dominant.
- I estimate the noise level and amplitude for the slit luminosity profile by comparing the original profile with a detrended version. I used a biweighted detrending algorithm with a windowing of 10 times the slit width, using the function `wotam.flatten` The choice of such a wide widow (about 40 px for a typical 100 µm slit size) allows us to have a good compromise to estimate both noise level and amplitude: Il fatto di usare la larghezza della slit come unità di misura per la scala spaziale è misleading, sarebbe meglio trovare qualcosa di piìnerente e magari meno arbitrario, per esempio usare la PSF
  - bkg level is the average value of the detrended profile. This is a good estimation of the real value since I used a large window and the light of the bright peaks is smeared on a wide area, making its contribution less relevant.
  - noise amplitude is the standard deviation of the original profile wrt the detrended one.

  Note choosing a wider window improves the estimation of the bkg level (even if it cannot actually ever converge to the real value) but increases the estimated noise amplitude.
- Cosmic rays and excessive noise are removed. In particular in the bluer part of the spectrum, due to the low sensitivity of the detector, after the flux calibration the noise is extremely high, seriously affecting the quality of the measurements. For example see the image below I removed these features by scanning each column of the frame looking for bright sharp peaks. Some good settings to remove both cosmic rays and noise is to find peaks very thin, with a width less than 3 px and an height of more than 5 times the average level of the bkg (i.e. the value estimated from the luminosity profile, divided by the number of pixels along the horizontal direction) bisognerebbe stimare la dimensione apparente dei raggi cosmici e confrontarla con la scala sul CCD per essere sicuri che funzioni con tutte le immagini.

  The peaks were identified with the function `scipy.signal.find_peaks` while their FWHM was estimated with `scipy.signal.peak_widths`. To be sure all the cosmic ray trace or the noise fluctuation was contained in the detected width I added a further 1 px on both directions along the columns.



- The columns that afeter the cleaning are depleted of more than the 25% their pixels are discarted as they are very likely to be noisy columns.
- On the cleaned frame I take again luminosity profile along the slit (i.e. integration on the wavelengths) to search for the astronomical sources. I used `scipy.signal.find_peaks` to find the position of the sources. I selected only those features with a width larger than 3 px (to be sure not to include unremoved cosmic rays) and with a prominence equal to the estimated noise amplitude.
- I computed the width of the sources with `scipy.signal.peak_widths`. In particular I considered the FWHM which proved to be more solid than the total width. Since many kinds of astronomical sources have a central core and bright wings, I decided to remove a that span $3.5 \times$ FWHM wrt the center of the peak to take into account such wings.
- At the end the script produces:
  - A plot of the luminosity profile along the slit (the one from the original frame, and after the cosmic ray removal) plus the masked source regions.
  - The bkg spectrum from the selected rows only, integrated along the spatial direction (plus the same for the full cleaned frame as a comparison).
  - A new file where cosmic rays, the UV noise and the astronomical sources are removed. The information in the hdr are preserved and the data when the file was produced is added.

A quantitative approach. I tried to sketch a qualitative model where I assumed gaussian profiles of the sources. The source ended when its flux was comparable to the amplitude of the bkg noise. The final distance Δ from the center of the source was

$$\Delta = \frac{1}{2\sqrt{2\ln 2}} \mathrm{FWHM} \sqrt{\ln(S_{\max}^2/B)}$$

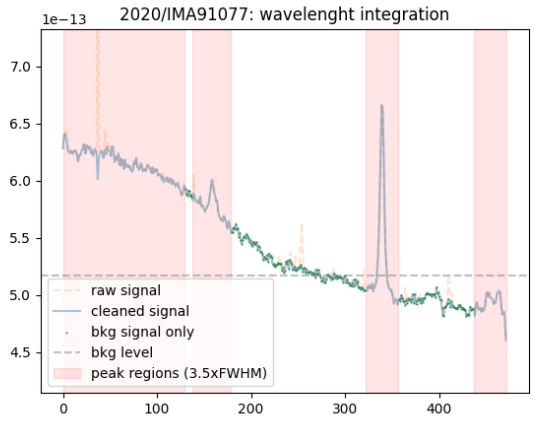
where $S_{\max}$ was the maximum counts of the source, while B the average level of bkg around the object. Since many extended objects cannot be reproduced by gaussian profiles this estimation may be misleading in many cases. Since the empirical procedure discussed above seems quite solid with current data, implementing a rigorous analytical approach that accounts for different luminosity profiles of the sources, is definitively unecessary.

2.2 Preliminary results

Here I report some relevant results I obtained.

All the data. Wavelength integrated profiles and bkg-removed spectra obtained with the current configuration are available in the directories `plots/integr` and `plots/sky_spec` respectively. The masked spectra are saved in the same directories of the original files with syntax `filename.fc.bkg.fits`.

IMA91077 (2020) and others. In this case we notice that the profile of the bkg is not uniform at all but seems to present a systematic trend. I need to investigate if it is intrinsic (e.g. due to a diffuse component) or the result of some calibration or instrumental issue. Similar features are presents in other frames like IMA61434 (2017), IMA84983, IMA85634, IMA91077 (2020), IMA95725 and IMA97134 (2021).



2.3 Comments on the first results

Frame limitations(?) I wonder whether 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 Bkg analysis

Once the bkg is extraced from each frame and saved into a new one, I can develope a script that reads the new filtered files and process them automatically.

4 Appendix: Python codes

4.1 Bkg extraction code

```
import numpy as np
import matplotlib.pyplot as plt
from astropy.io import fits
from scipy.signal import find_peaks, peak_widths
from datetime import datetime
import glob
import os
from wotan import flatten

########################
#OPTIONS
savefiles = True
plot_profile = True
plot_spec = True
show_ima = False

#PARAMS
data_col_frac = .75 #minimum fraction of valid pixels in a column
width_mult = 3.5 # interval to exclude around a source, wrt center in FWHM units
cr_pad = 1 # number of px to exclude around cr, in a fixed column
LAMBDA_lim = 3500 #A, limit blue wavelength
########################

#browse all the *.fc.fits files in a directory and its subdirectories
#main_path = './Asiago_nightsky/2006/'
main_path = './'
file_ls = glob.glob(main_path+'/**/*.fc.fits', recursive= True)
names = [os.path.basename(x) for x in file_ls]

#initialize the .log file
f = open("bkg_extr.log", "w")
f.write('Running bkg_extr.py at '+
        datetime.now().strftime("%H:%M:%S, %Y-%m-%d")+'\n')
f.close()
warnings_count = 0

########################
########################

#process all the files found
for name,file in zip(names,file_ls):

    print('processing file '+name+'\n', end='\r')

    #open a FITS file
    hdul = fits.open(file)
    hdr = hdul[0].header

    #extract wavelenght information from the header
    NAXIS1, NAXIS2 = hdr['NAXIS1'], hdr['NAXIS2']
    LAMBDA0, DELTA = hdr['CRVAL1'], hdr['CDELT1']

    LAMBDA = np.arange(LAMBDA0, LAMBDA0+NAXIS1*DELTA, DELTA)
    if len(LAMBDA) == NAXIS1+1:
        LAMBDA = LAMBDA[:-1]

    #remove extreme blue wavelengths
    LAMBDA_start_id = 0
    if LAMBDA0 <= LAMBDA_lim:
        LAMBDA_start_id = len(LAMBDA)-len(LAMBDA[ LAMBDA>LAMBDA_lim])
        LAMBDA = LAMBDA[LAMBDA_start_id:]

    year = hdr['DATE-OBS'][:4]

    #aperture information from the hdr
    SLIT = hdr['SLIT'] #microns
    try:
        BINX, BINY = hdr['BINX'], hdr['BINY']
        TELSCALE = hdr['TELSCALE'] #arcsec/mm
        CCDSCALE = hdr['CCDSCALE'] #arcsec/px
    except KeyError:
        BINX, BINY = hdr['HBIN'], hdr['VBIN']
```

```
        warnings_count += 1
        no_scale = ' WARNING: no scale info in the hdr (using defauls)'
        f = open("bkg_extr.log", "a")
        f.write(file+no_scale+'\n')
        f.close()

        TELSCALE = 10.70 #arcsec/mm #TO BE CHECKED!!!
        CCDSCALE = 0.60 #arcsec/px #TO BE CHECKED!!!

    SLIT_angular = SLIT/1000 * TELSCALE #slit size in arcsec
    SLIT_px = SLIT_angular / CCDSCALE / BINX #slit size in px

######################
    #bkg level estiamtion
    raw_data = hdul[0].data[:,LAMBDA_start_id:]
    raw_integr = np.sum(raw_data, axis = 1)

    x = np.arange(len(raw_integr))

    #signal detrend
    flat,trend = flatten(
        x,
        raw_integr,
        method='biweight',
        window_length=10*SLIT_px,
        cval = 1, return_trend = True)

    #esimate the bkg level and noise amplitude
    bkg_est = np.nanmean(trend)
    noise = np.nanstd(raw_integr-trend)

######################
    #remove cosmic rays and UV noise
    data = np.copy(raw_data)

    cr_col_frac = np.zeros(len(LAMBDA)) #fraction of remaining px
    for cr_col,col in enumerate(data.T):
        cr_line,_ = find_peaks(col,
                               prominence = bkg_est/len(raw_data[1])*5,
                               width = (0,3))

        cr_widths = peak_widths(col, cr_line, rel_height=0.5)[0]

        #set left and right boundaries of the source region along the slit
        left_width = cr_line-cr_widths - cr_pad
        right_width = cr_line+cr_widths + cr_pad

        #scan each column and remove peaks
        cr_sel = np.zeros(np.shape(col), dtype=bool)
        for i in range(np.shape(col)[0]):
            for peak,width in zip(cr_line,cr_widths):
                if abs(i-peak) < width+cr_pad:
                    cr_sel[i] = True

        #counts how many pixels are left in a column
        saved_px = (NAXIS2 - np.sum(cr_sel))/NAXIS2
        cr_col_frac[cr_col] = saved_px
        if saved_px >= data_col_frac: #if enough, take the masked column
            data[cr_sel, cr_col] = np.nan
        else: #else discart the entire column
            data[:, cr_col] = 0.

######################
    #use noise/bkg info to find peaks
    integr = np.nansum(data, axis = 1)

    peaks,properties = find_peaks(integr, prominence=noise, width = 3)
    peak_FWHM = peak_widths(integr, peaks, rel_height=0.5)[0]

    #set left and right boundaries of the source region along the slit
    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
    try:
        if left_width[0] < 0:
            left_width[0] = 0
        if right_width[-1] > NAXIS2:
            right_width[-1] = NAXIS2
    except IndexError:
        no_source = " WARNING: no sources were detected"
        warnings_count += 1
        f = open("bkg_extr.log", "a")
        f.write(file+no_source)
        f.close()

########################
    #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 luminosity profile, show source and bkg regions
    if 1 == plot_profile:
        plt.title(year+'/'+name[:-8]+': wavelenght integration')
        plt.plot(raw_integr, alpha=0.2, ls='dashed', c='C1')
        plt.plot(integr, alpha=0.4) #integrated flux
        plt.scatter(x[bkg_sel],
                    integr[bkg_sel],
                    s=0.2, c='green') #select bkg

        #esimate the bkg of the filtered regions only
        bkg_est_filt = np.mean(integr[bkg_sel])

        plt.axhline(y=bkg_est_filt, ls='dashed', c='grey', alpha=0.5)
        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 settings
        plt.ylim(min(integr)*0.9, max(integr)*1.1)
        plt.legend(['raw signal','cleaned signal',
                    'bkg signal only', 'bkg level',
                    f'peak regions ({width_mult}xFWHM)'])

        if savefiles is True:
            plt.savefig('./plots/integr/'+year+'_'+name[:-8]+'.png')
            plt.close()
        else:
            plt.show()

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

    #plot the spatially integrated spectrum of the bkg
    if 1 == plot_spec:
        plt.title(year+'/'+name[:-8]+': bkg spectrum')
```

```
        plt.plot(LAMBDA, total, label='full frame', color='gray', alpha=0.3)
        plt.plot(LAMBDA, sky, label='sky only')
        plt.legend()
        if savefiles is True:
            plt.savefig('./plots/sky_spec/'+year+'_'+name[:-8]+'.png')
            plt.close()
        else:
            plt.show()

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

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

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

######################
    #save masked data in a new FITS file
    if 1 == True:
        now = datetime.now()
        now_str = now.strftime("%Y-%m-%d %H:%M:%S")

        hdr.set('BKGEXTR', now_str, 'Time of bkg extraction')
        hdr.set('UVLIM', LAMBDA_lim, 'A')
        hdr['NAXIS1']=len(data[0])
        new_hdu = fits.PrimaryHDU(ma_data)
        new_hdul = fits.HDUList([new_hdu])
        new_hdul[0].header = hdr

        file_new = file[:-5]+'.bkg.fits'
        new_hdul.writeto(file_new, overwrite=True)

if warnings_count != 0:
    print(f'WARNING: {warnings_count} warnings occurred (see the log)')
```

4.2 Bkg analysis code

```
import numpy as np
import numpy.ma as ma
import matplotlib.pyplot as plt
from astropy.io import fits
import astropy.coordinates as coord
from astropy import units as u
from scipy.signal import find_peaks, peak_widths, detrend
from scipy import ndimage
from datetime import datetime
import glob
import os
from wotan import flatten

######################
#OPTIONS
savefiles = False
plot_profile = False
plot_spec = False
show_ima = False

#PARAMS

######################

#browse all the *.fc.fits files in a directory and its subdirectories
main_path = './Asiago_nightsky/2006/'
#main_path = './'
file_ls = glob.glob(main_path+'/**/*.fc.bkg.fits', recursive= True)
names = [os.path.basename(x) for x in file_ls]

#process all the files found
```

```
for name,file in zip(names,file_ls):

    #load the frame
    hdul = fits.open(file)
    hdr, data = hdul[0].header, hdul[0].data

    #take wavelength info from the hdr
    NAXIS1, NAXIS2 = hdr['NAXIS1'], hdr['NAXIS2']
    LAMBDA0, DELTA = hdr['CRVAL1'], hdr['CDELT1']
    LAMBDA_lim = hdr['UVLIM']

    #the (eventually) UV-limited wavelengths array
    LAMBDA = np.arange(LAMBDA_lim, max(LAMBDA_lim, LAMBDA0)+NAXIS1*DELTA, DELTA)

    '''
    HERE IT GOES THE CODE TO ANALYZE THE FEATURES OF BKG SPECTRA
    '''
```