Notes

Marco Codato

June 27, 2022

1 Introduction

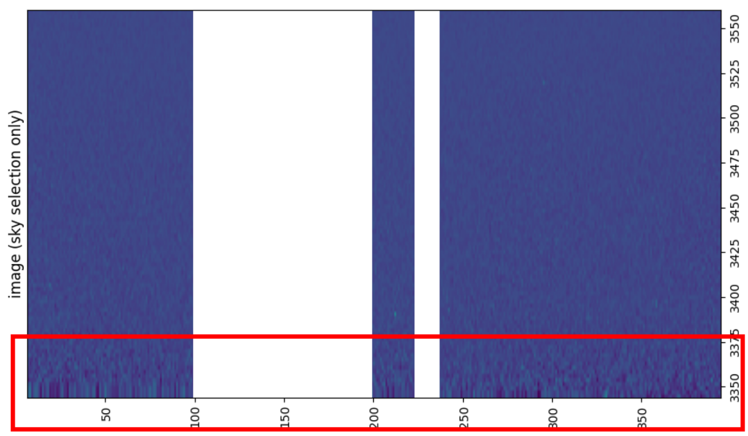
Moved to the GitHub repository `codatomrc/tesi`.

2 Background spectrum extraction

2.1 Automatic detection

The script I am writing works as follows.

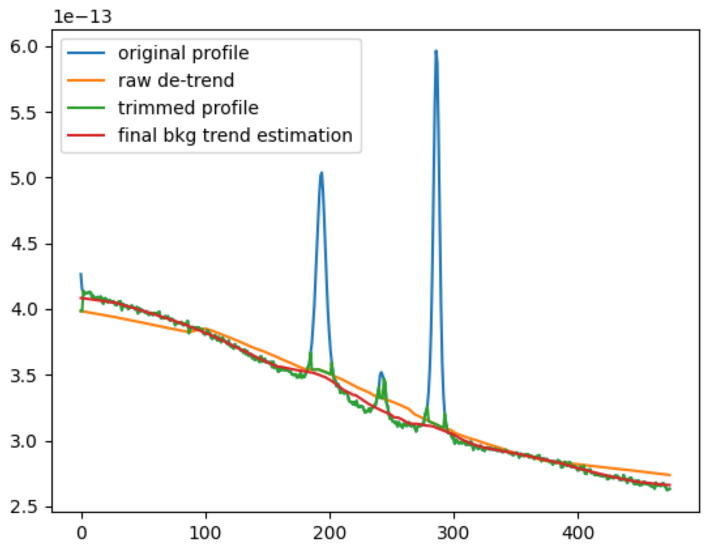
- Automatically explores a directory and find all the raw files, marked as `filename.fc.fits`. All the files are then processed one by one.
- Far each frame relevant information is extracted from the header (hdr). In particular the script requires
  - information about the wavelength of each pixel (initial 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 wavelengths 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 preliminary decided to cut blue wavelengths at 3500 Å. This value was bull-eyed looking at bkg spectra and realizing that this is the threshold where the noise becomes dominant.
- 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 after the cleaning are depleted of more than the 25% their pixels are discarded 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. Some of the frames present a systematic trend in the bkg which makes the source detection not trivial. I decided to proceed as follows:
  1. Estimate the general trend of the luminosity profile. I used a bi-weighted detrending algorithm with a windowing of 200 px and the fine-tuning parameter `cval=10`, using the function `wotam.flatten`. Such large window size is still not enough to cancel out the effect of the brightest features.
  2. Trim the original light profile by removing all the data that emerges from the bkg global trend. I used as threshold the detrended profile, vertically shifted by a factor equal to the 5% of the average bkg level. A good estimator for the avg bkg level turned to be the median of the original light profile.
  3. Perform a new detrend on the trimmed data. In this case I used a much narrower windows size, or 50 px since it was not necessary to dilute the bright peaks in correspondence of the sources. This can be considered as an estimator for the global bkg trend along the slit.



- The source peaks are detected automatically with `scipy.signal.find_peaks`. Peaks were identified after removing the contribution of the systematic bkg trend detected in the previous step. I selected only those features with a width larger than 3 px (to be sure not to include residuals cosmic rays) and an height greater than the 5% of the bkg (median) level.
- 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 $2.5 \times$ FWHM wrt the center of the peak to take into account such wings. In some cases the luminosity profile of the sources is not Gaussian and the tails are brighter than the Gaussian prevision. In these cases I considered as related to a source also all the pixels around the peak that are brighter than what expected from the detrended profile.
- 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 not necessary.

2.2 Comments on the first results

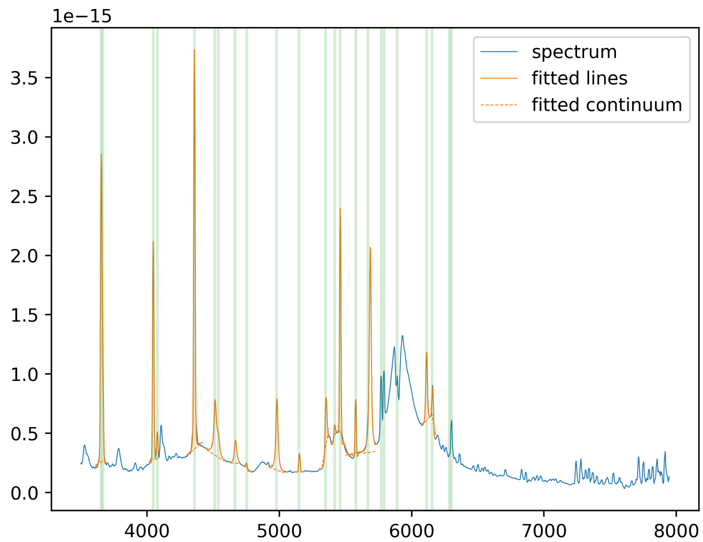
Frame limitations(?) I wonder whether the whole CCD area is suitable for collecting data or it would be better/necessary to neglect some specific regions, both in the spatial and dispersion directions.

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

3 Bkg analysis

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

3.1 Line analysis



I want to measure most of the spectral lines contained in sky spectra and reported in the table `tesi/lines.txt`. I particular I am interested in the equivalent width (EW). Eventually I also shall try to identify further lines, e.g. the bright one at ~ 3800 Å. The procedure I plan to follow is the following

- Average all the px in each column in the original 2D frame to get the bkg uni-dimensional spectrum.
- Identify the positions of the known lines (table `lines.txt`) in the spectra. Note that due to the low resolution of the frames, some lines are totally unresolved. I will consider as single unresolved lines, those with a wavelength difference lower than 2 times the pixel resolution (which is of the order of ~ 3 Å).
- Identify the continuum below the lines by using a double median filter as done in the bkg extraction, i.e. by using the `wotam.flatten` function. For the preliminary filtering I used a window length of $200\Delta\lambda$ (with $\Delta\lambda$ the wavelengths size of each pixel). I used this initial function, augmented by the 30% of the median value of the spectrum, as threshold for the data to be considered in the final detrended profile.

- I fitted the listed lines with Gaussian profiles using the `specutils.fitting.fit_lines` utility. Guess position of each line was the theoretical position, while guess amplitude was half of the peak value of the spectrum and dispersion was fixed at 5 Å.

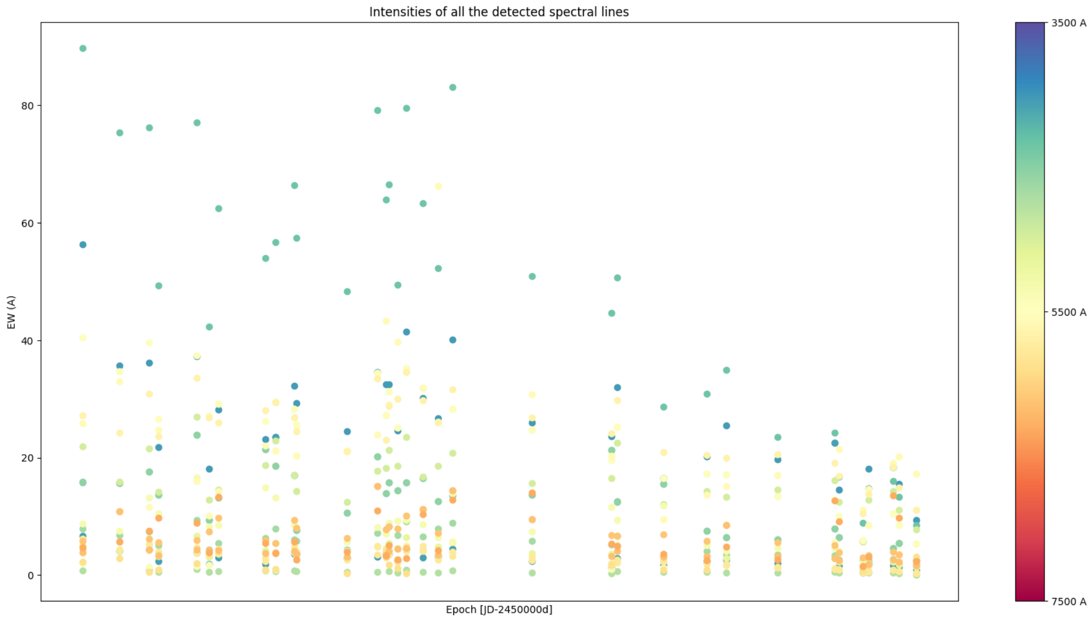
- EWs are computed as

$$\mathrm{EW} = \sum_i \frac{L(\lambda_i, \vec{p}) - C(\lambda_i)}{C(\lambda_i)}$$

with L and C respectively the fluxes of the best fit line (characterized by the params $\vec{p}$) and the continuum. With this definition, emission lines have positive widths.

3.1.1 Tentative results

I can plot the resulting EWs as a function of the epoch of observation, as shown below. We can notice that in



general red lines (to be identified, maybe Na?) are weaker in the most recent spectra. Concerning bluer lines apparently is seems the same as the red ones, but one should check more quantitatively.

A possible explanation is that this is due to the reduction of Na/Hg street lamps in favor of LED lights (that have no lines but a continuum emission).

3.1.2 Known issues

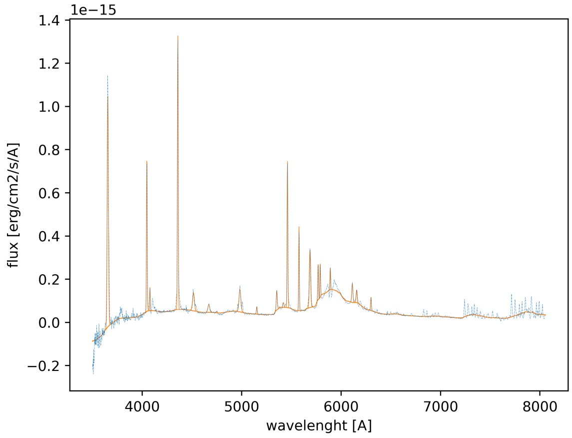
Residual calibration uncertainties. In one of the first plot, in the frame `ima_010` (2006) the spectrum in the blue side reaches negative flux counts. This is probably due to the poor calibration accuracy at such extreme wavelengths (see below).

Solidity of the fit. In some cases, where several lines are confined in few Å, the script may fit the depression between two lines insted of the actual line. Currently those results are rejected, it would be ideal if a could manage to perform the fit correctly, instead.

Fidelity of the continuum. In the figure of example above one can notice that my estimation of the continuum is not always totally exact. In some situations my estimation is clearly above or below the desired level:

- In the presence of broad separated lines and a noisy continuum (e.g. iu the bluer side) the algorithm may interpret everything as continuum and average the lines levels. Note in these cases the spectrum is also hevely affected by noise and such counts are not very reliable.

- In the presence of complex structures like those at $\sim 589\,\mathrm{nm}$ (see next paragraph) the continuum is not correctly identified.

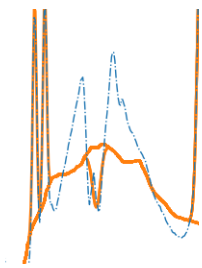


- Although the script try to remove the contribution of broad strong lines by cutting the peaks in some cases the tails are still bright enough to affect the final result, in particular where there are groups of bright lines, where the tails overlaps and occupy a broad region of the spectrum. See for example the feature at $\sim 550\,\mathrm{nm}$. In these cases the continuum is overestimated and thus for very bright lines we expect to have a slightly underestimated EW.

The 5889/5895 Na doublet lines. These lines are located in a critical position where 3 different effects overlaps:

1. a broad absorption band, over
2. a sharp (artificial) emission peak, everything on top of
3. the maximum of the artificial light emission continuum.

For such a lot of factor the measurements of the emission line is extremely challenging. Note also the estimation of the continuum level, necessary for the EW, is very uncertain. Probably one of the weakest point of my method for extrapolating the spectrum continuum below the lines is the difficult to fit such feature, as shown in the detail below. In such particular case the estimated continuum severely departure from the real one plus



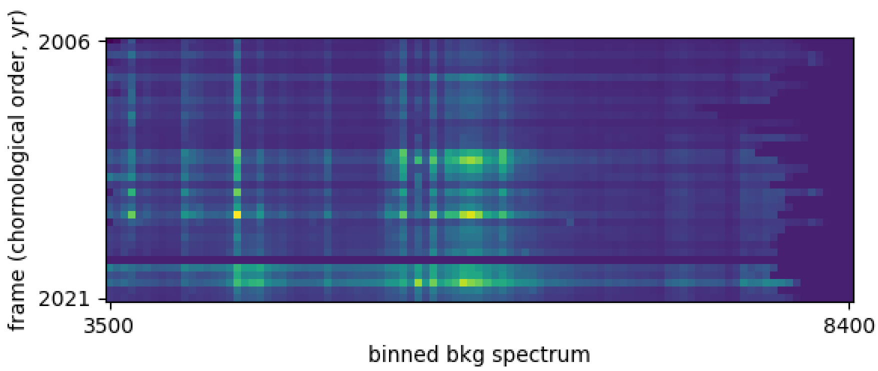
it is not correctly recognized the line to be fitted as the emission one.

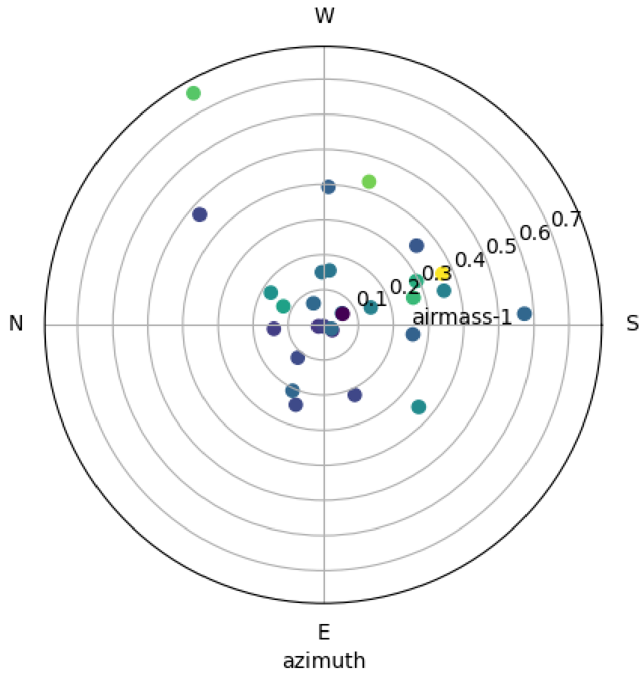
3.2 (Tentative) analysis on the whole spectra

Date correlation. Bkg spectra are characterized by few very bright emission lines plus one/two continuum region(s). A very first analysis may consist on lower the resolution of the spectra and compare them with each others. In particular I decided to bin them in 100 Å bin size and chronologically order them. Data seems to be consistent with the claim that the bkg level has increased in the years, likely due to artificial light.

Orientation correlation. I plotted the position of each frame on the sky sphere (in terms of airmass and azimuth) against the total luminosity of each spectrum (i.e. integrated in the wavelengths). Apparently brightest measurements are concentrated toward the S/W direction.

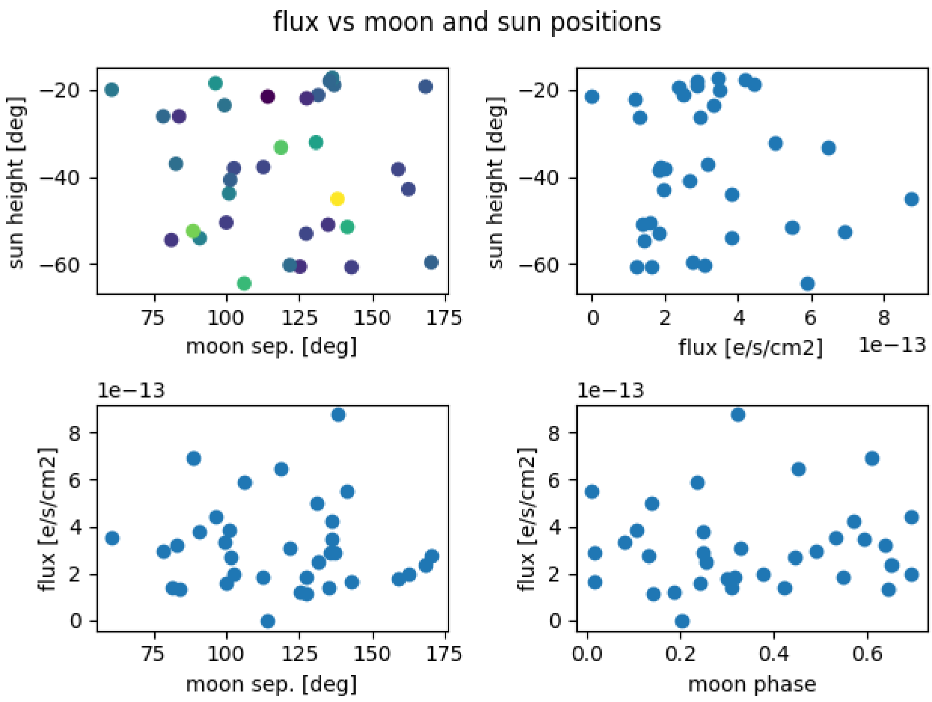
Moon and Sun position correlations. I checked whether the position of Moon and Sun affected the total bkg counts. The quantities I checked against the total flux are the Sun altitude below the horizon, the angular separation of from the moon and the moon phase at the epoch of observation. No evident trends emerged

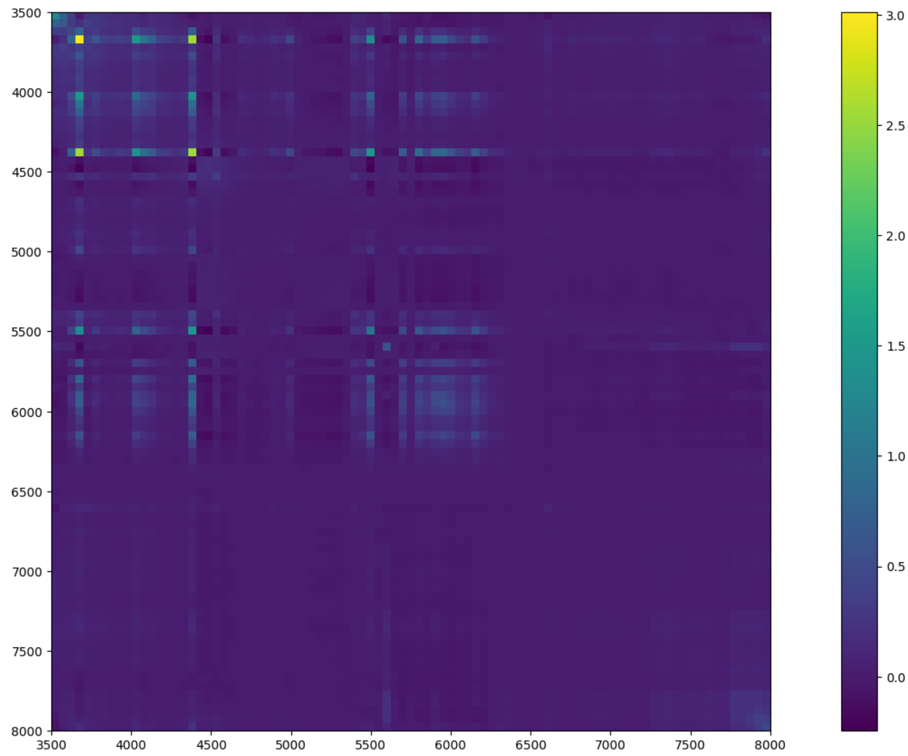




from this analysis, although this is to the smartest choice for the plotted quantities. For example all of the 3 quantities of the plot must be considered at the same time since their effects should sum up.

Spectral features correlation. I checked for correlations in the binned spectra by computing the covariance matrix of binned data. I normalized the spectra with their value on the 58-th bin, i.e. at $\sim 6450\,\text{Å}$, a region of the bkg spectrum where no relevant continuum nor evident lines are present. Resulting correlations have to be further investigated.





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
```

```python
from wotan import flatten
from scipy.optimize import curve_fit

######################
#OPTIONS
save_plots = True
save_FITS = False
plot_profile = True
plot_spec = True
show_ima = False

#PARAMS
peak_height = 0.05 #height abouve the bkg level
data_col_frac = .75 #minimum fraction of valid pixels in a column
width_mult = 2 # interval to exclude around a source, wrt center in FWHM units
cr_width = 2.5 # cr trace spatial width
cr_prominence = 5 #treshold height wrt average column level
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/2020/'
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()

#import table of known lines
lines = np.genfromtxt('lines.txt', usecols=0)

#append to the log
warnings_count = 0
def log_append(message):
    f = open("bkg_extr.log", "a")
    f.write(message+'\n')
    f.close()

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

#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']

    #generate the lambdas array
    if hdr['CTYPE1'] != 'LINEAR':
        log_append('WARNING: no linear wavelength calibration')
    LAMBDA = np.arange(LAMBDA0, LAMBDA0+NAXIS1*DELTA, DELTA)
    if len(LAMBDA) == NAXIS1+1:
        LAMBDA = LAMBDA[:-1]

    #remove extreme blue wavelengths
    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']

        log_append(' WARNING: no scale info in the hdr (using defauls)')

        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))

    bkg_est = np.nanmedian(raw_integr)

######################
    #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):
        col_avg = np.nanmean(data[:,cr_col])
        cr_line,_ = find_peaks(col,
                               prominence = cr_prominence*col_avg,
                               width = (0,cr_width))

        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)
    bkg_est = np.median(integr)

    #detrend: global trend (including peaks)
    _,trend_raw = flatten (x,
                           integr ,
                           method ='biweight',
                           window_length =200 ,
                           cval = 10, return_trend = True )

    #trim removing peaks, i.e. data fare above the global trend
    integr_trim = np.where(integr <= trend_raw+0.05*bkg_est,
                          integr, trend_raw)

    #detrend the trimmed data, much less sensitive to the peaks
    _,trend = flatten (x,
                       integr_trim ,
                       method ='biweight',
                       window_length =50 ,
                       cval = 10, return_trend = True )
    '''
```

```
    more plot about detrending
    plt.plot(x,integr, label='original profile')
    plt.plot(x,trend_raw, label='raw de-trend')
    plt.plot(x,integr_trim, label='trimmed profile')
    plt.plot(x,trend, label='final bkg trend estimation')
    plt.legend()
    plt.show()
    '''

    #detrend residuals: original peaks are highlighted wrt the bkg profile
    diff = integr-trend

    #find peaks
    peaks,properties = find_peaks(diff, height=0.05*bkg_est, width = cr_width)
    peak_FWHM = peak_widths(integr, peaks, rel_height=.5)[0]/2.

    if len(peak_FWHM)== 0:
        no_source = " WARNING: no sources were detected"
        log_append(no_source)

    #generate a boolean mask True outside the peaks
    bkg_sel = np.full(np.shape(x), True)
    for i,peak in enumerate(peaks):
        width = (int(peak_FWHM[i])+1)*width_mult
        for w in range(-width,width):
            bkg_sel[peak+w]=False
        w = width -1
        while integr[peak+w] >= trend[peak+w]:
            bkg_sel[peak+w] = False
            w += 1
        w = width
        while integr[peak-w] >=trend[peak-w]:
            bkg_sel[peak-w]=False
            w += 1

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

    #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.plot(x, trend_raw+0.05*bkg_est, ls='dashed', c='grey', alpha=0.5)
        ima = np.zeros(np.shape(bkg_sel))
        plt.fill_between(x, ~bkg_sel*1.1*max(integr), color='red', alpha=.1)

        #plt settings
        plt.ylim(min(integr)*0.9, max(integr)*1.1)
        plt.legend(['raw signal','cleaned signal',
                    'bkg signal only', 'detrended treshold',
                    f'peak regions ({width_mult}xFWHM)'])

        if save_plots 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

    total[total == 0] = np.nan

    #plot the spatially integrated spectrum of the bkg
    if 1 == plot_spec:
        plt.title(year+'/'+name[:-8]+': bkg spectrum')
        plt.plot(LAMBDA, total,  color='gray', alpha=0.3)
```

```
        plt.plot(LAMBDA, sky)
        for line in lines:
            plt.axvline(x=line, c='C1', alpha=.2)
        plt.legend(['full frame', 'sky only', 'known lines'])
        if save_plots 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) and (save_plots ==0):
        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 == save_FITS:
        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)

f = open("bkg_extr.log", 'r')
warnings_count = len(f.readlines())-1
if warnings_count != 0:
    print(f'WARNING: {warnings_count} warnings occurred (see the log)')
```

4.2 Bkg analysis code

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.signal import find_peaks, peak_widths
from astropy.io import fits
from astropy import units as u
from astropy.modeling import models
from astropy.table import Table
from datetime import datetime
import glob
import os
from wotan import flatten
from specutils.fitting import fit_lines
from specutils.spectra import Spectrum1D
######################
#OPTIONS

plot_fits = True
save_fits = True

FITS_lines = False
#PARAMS
line_res = 2 #x delta lambda,
JD0 = 2450000
######################

# import lines table
lines_raw = np.genfromtxt('lines.txt', usecols=0)
line_diff = np.diff(lines_raw)
```

```
widths = []
JDs = []

#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
file_id = 0
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']
    year = hdr['DATE-OBS'][:4]
    JDs.append(hdr['JD'])

    #the (eventually) UV-limited wavelengths array
    LAMBDA_start = max(LAMBDA_lim, LAMBDA0)
    LAMBDA = np.arange(LAMBDA_start, LAMBDA_start+NAXIS1*DELTA, DELTA)
    if len(LAMBDA) == NAXIS1+1:
        LAMBDA = LAMBDA[:-1]
    spec = np.nanmean(data, axis=0)

    #remove blended lines, i.e. to be considered as a single feature
    close_lines = np.where(line_diff < line_res*DELTA, False, True)
    close_lines = np.insert(close_lines, 0, True)
    lines = lines_raw[close_lines]

    filename = './plots/widths/'+year+'_'+name[:-13]+'.l.txt'
    f = open(filename, 'w') if FITS_lines else 0
    f.write(f"#line\t EW") if FITS_lines else 0

    '''
    LINE FIT
    '''
    #continuum estimation
    lvl_est = np.median(spec)
    _,trend_raw = flatten (LAMBDA,
                           spec ,
                           method ='biweight',
                           window_length =200 ,
                           cval = 10, return_trend = True )

    #trim removing peaks, i.e. data far above the global trend
    spec_trim = np.where(spec <= trend_raw+0.3*np.median(spec), spec, trend_raw)

    #detrend the trimmed data, much less sensitive to the peaks
    _,trend = flatten (LAMBDA,
                       spec_trim ,
                       method ='biweight',
                       window_length =150 ,
                       cval = 5, return_trend = True )

    #line fit and EW computation
    u_flux = u.erg / (u.cm ** 2 * u.s * u.AA) #flux units
    A = u.AA #angstrom units
    spectrum = Spectrum1D(flux=spec*u_flux, spectral_axis=LAMBDA*A)
    EWs = []

    for line in lines:
        line_init = models.Gaussian1D(amplitude=0.5*max(spec)*u_flux,
                                      mean=line*A,
                                      stddev=5.*A)

        line_fit = fit_lines(spectrum-trend, line_init)
        y_fit = line_fit(LAMBDA*A)

        plt.plot(LAMBDA, y_fit+trend*u_flux,
```

```
                 lw=0.4, ls = '-', c='C1') if plot_fits else 0

        EW = np.sum(y_fit/(trend*u_flux))
        EWs.append(EW)

        f.writelines(f"\n{line}\t {EW}") if FITS_lines else 0


    if plot_fits is True:
        plt.plot(LAMBDA, spec, lw=0.2, ls='-.')
        plt.xlabel('wavelenght [A]')
        plt.ylabel('flux [erg/cm2/s/A]')
    if (save_fits is True) and (plot_fits is True):
        plt.savefig('./plots/line_fit/'+year+'_'+name[:-8]+'.png', dpi=500)
    elif plot_fits is True:
        plt.show()
    plt.close()

    #plt.plot(EWs, '-o')
    f.close() if FITS_lines else 0

    if FITS_lines is True:
        #save new FIT file with with EW in a partition
        table_hdu = fits.BinTableHDU.from_columns(
            [fits.Column(name = 'line', array = lines, format = 'E'),
             fits.Column(name = 'EWs', array = EWs, format = 'E')])

        now = datetime.now()
        now_str = now.strftime("%Y-%m-%d %H:%M:%S")

        hdul.append(table_hdu)
        t_hdr = hdul[1].header
        t_hdr.set('UNITS', 'Angstrom')
        t_hdr.set('EWTIME', now_str, 'Time of EW computation')


        x = file[:-12]+'.l.bkg.fits'
        hdul.writeto(file_new, overwrite=True)



#plt.show()
widths = np.asarray(widths)

#remove bad fits
neg_sel = widths <= 0.
widths[neg_sel] = np.nan


cm = plt.cm.Spectral(np.linspace(1, 0, 7500-3500))
for i in range(len(widths.T)):
    plt.plot(JDs, widths.T[i], 'o', color=cm[int(lines[i])-int(3500)])
plt.xlabel(f'Epoch [JD-{JD0}d]')
plt.ylabel('EW (A)')
#plt.ylim(0,+50)
sm = plt.cm.ScalarMappable(cmap='Spectral')
cbar = plt.colorbar(sm, ticks=[0,0.5,1])
cbar.set_ticklabels(['7500 A','5500 A','3500 A'])
plt.title('Intensities of all the detected spectral lines')
plt.xticks([])
plt.show()
```