Quentin_Trombini_Week13

S13_01_a

```
import matplotlib.backends.backend_agg as agg
import photutils.background as background
import astroquery.simbad as simbad
import astropy.coordinates as coord
import astropy.units as units
import astroquery.skyview as skyview
import astropy.wcs as wcs
import matplotlib.backends.backend_agg as agg
import matplotlib.figure as figure
import astropy.stats as stats
import astropy.table as table
import numpy as np
import astropy
import photutils
import astropy
import os
import matplotlib
import astroalign

# ---------------------- RETRIEVE AND DOWLOAD STAR DATA -------
if not os.path.isfile("./S13/optical.fits") or not os.path.isfi
    query_result = simbad.Simbad.query_object("M31")

    RA = query_result['RA']
    Dec = query_result['DEC']

    coordinates = coord.SkyCoord(RA[0], Dec[0], unit=(units.hou
    image = skyview.SkyView.get_images(position=coordinates, su
    astropy.io.fits.writeto("./S13/optical.fits",image[0][0].da
```

```
    ir_ra_decimal = coordinates.ra.deg + 1 /60
    ir_dec_decimal = coordinates.dec.deg + 2 /60

    ir_coordinates = coord.SkyCoord(ir_ra_decimal,ir_dec_decimal
    ir_image = skyview.SkyView.get_images(position=ir_coordinate
    astropy.io.fits.writeto("./S13/IR.fits",ir_image[0][0].data,

# ---------------------- CREATE AND PRINT OPTICAL IMAGE --------
    with astropy.io.fits.open("./S13/optical.fits") as hdu_list
        header = hdu_list[0].header
        wcs_head= wcs.WCS(header)
        image = hdu_list[0].data

    fig= matplotlib.figure.Figure()
    canvas = agg.FigureCanvasAgg(fig)
    ax = fig.add_subplot(111, projection=wcs_head)

    norm = astropy.visualization.mpl_normalize.ImageNormalize( s
    im = ax.imshow(image, origin='lower', cmap="grey", norm=norm
    fig.savefig("./S13/optical.png", dpi=150)
    fig.clf()

# ------------------------- CREATE AND PRINT IR IMAGE ----------
    with astropy.io.fits.open("./S13/IR.fits") as hdu_list:
        header = hdu_list[0].header
        wcs_head= wcs.WCS(header)
        image = hdu_list[0].data

    canvas = agg.FigureCanvasAgg(fig)
    ax = fig.add_subplot(111, projection=wcs_head)

    norm = astropy.visualization.mpl_normalize.ImageNormalize( s
    im = ax.imshow(image, origin='lower', cmap="grey", norm=norm
    fig.savefig("./S13/IR.png", dpi=150)

# --------------------------- CATALOGUE CREATION --------------
```

```
def catalogue_creation(input):
    sigma_sky = 3.0
    maxiters = 10
    box_size = 50
    filter_size = 3
    thresh = 2
    kernel = 3.0
    array_size = 5
    nb_pixel = 10
    nb_level = 32
    vontrast = 0.001

    with astropy.io.fits.open("./S13/"+input+".fits") as hdu:
        header = hdu[0].header
        image = hdu[0].data
        if(header['NAXIS'] == 0):
            header = hdu[1].header
            image = hdu[1].data
    sigma_clip = stats.SigmaClip(sigma=sigma_sky, maxiters=maxit
    skybg_estimator = background.SExtractorBackground()

    image_skybg = background.Background2D(image, box_size=(box_s
    image_skysub = image - image_skybg.background
    detection_threshold = thresh * image_skybg.background_rms_me
    convolution_kernel = photutils.segmentation.make_2dgaussian_
    image_convolved = astropy.convolution.convolve(image_skysub,
    image_segmented = photutils.segmentation.detect_sources(imag
    catalogue = photutils.segmentation.SourceCatalog(data=image_
    table_source = catalogue.to_table()
    astropy.io.ascii.write(table_source, "./S13/"+input+".cat",

if not os.path.isfile("./S13/optical.cat") or not os.path.isfile
    catalogue_creation("optical")
    catalogue_creation("IR")

# -------------------------- LOOKING FOR COMMON STAR ----------
```

```
table1 = table.Table.read("./S13/optical.cat",format='ascii.comm
table2 = table.Table.read("./S13/IR.cat",format='ascii.commente

list_source1_x = list (table1['xcentroid'])
list_source1_y = list (table1['ycentroid'])
list_source2_x = list (table2['xcentroid'])
list_source2_y = list (table2['ycentroid'])
position_1= np.transpose ( (list_source1_x, list_source1_y) )
position_2= np.transpose ( (list_source2_x, list_source2_y) )

# finding star-to-star matching
transf, (list_matched_2, list_matched_1) = astroalign.find_trans
list_matched_2_aligned = astroalign.matrix_transform (list_matcl

# ------------------------------ STAR ALIGNEMENT ---------------
# --------- THIS PART IS COMING FROM THE COURSE SINCE IN COULDNT

(header1, image1) = astropy.read_fits("./S13/optical.fits")
(header2, image2) = astropy.read_fits("./S13/IR.fits")

image1 = image1.byteswap ().newbyteorder ()
image2 = image2.byteswap ().newbyteorder ()



st = coord.skimage.transform.SimilarityTransform (scale=transf.s
image2_aligned = coord.skimage.transform.warp (image2, st.invers

markers = ['o', 'v', '^', 's', 'p', 'h', '8']
colours = ['maroon', 'red', 'coral', 'bisque', 'orange', 'wheat

fig = figure.Figure ()
agg.FigureCanvasAgg (fig)
ax1 = fig.add_subplot (121)
ax2 = fig.add_subplot (122)

norm1 = astropy.visualization.mpl_normalize.ImageNormalize ( str
```

```
im1 = ax1.imshow (image1, origin='lower', cmap='bone', norm=norr
for i in range ( len (list_matched_1) ):
    i_marker = i % len (markers)
    i_colour = i % len (colours)
    ax1.plot (list_matched_1[i][0], list_matched_1[i][1], marke
ax1.set_title ('First Image')


norm2 = astropy.visualization.mpl_normalize.ImageNormalize ( str
im2 = ax2.imshow (image2_aligned, origin='lower', cmap='bone', r
for i in range ( len (list_matched_2_aligned) ):
    i_marker = i % len (markers)
    i_colour = i % len (colours)
    ax2.plot (list_matched_2_aligned[i][0], list_matched_2_alig
ax2.set_title ('Second Image')
fig.tight_layout ()
fig.savefig ("./S13/final.png", dpi=150)
```

S13_01_b

```
Exception has occurred: MaxIterError ×
List of matching triangles exhausted before an acceptable transformation was found
  File "/home/Clamore/Documents/NCU/Astro/S13/S13_01.py", line 114, in <module>
    transf, (list_matched_2, list_matched_1) = astroalign.find_transform (position_2, position_1)
                                               ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
astroalign.MaxIterError: List of matching triangles exhausted before an acceptable transformation was found
```

S13_01_c

Retrieve and download data

```
if not os.path.isfile("./S13/optical.fits") or not os.path.isfi
```

All of this part is done inside this if condition to avoid loosing time when I try different things after.

First of I try to query my star data using simbad.Simbad.query object()

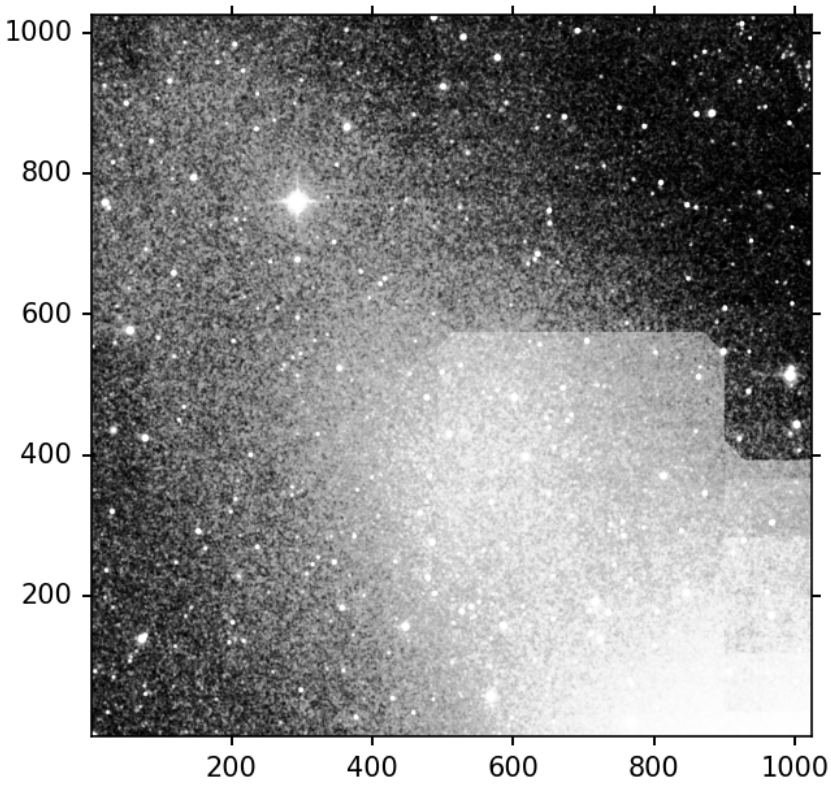
once this is done I extract the coordinate from the query result using astropy.coordinate.Skycoord

Then I will add the offset to the coordinate and dowload the two image of the sky: the first one using SDSSr to get an optical version, the second one using 2MASS-J to get the near-infrared version.

those data will be converted and stored as fits file

Print image

Once i have the fits data I will use matplolib to print the image to see if there is a problem as you can see the image below is really blurry so my data must have been wrongly downloaded



Catalogue creation

To use the alignement I need a catalogue i will crate it from some data and the image previously collected:

```
image_skybg = background.Background2D(image, box_size=(box_size,
image_skysub = image - image_skybg.background
detection_threshold = thresh * image_skybg.background_rms_median
convolution_kernel = photutils.segmentation.make_2dgaussian_kern
image_convolved = astropy.convolution.convolve(image_skysub, con
image_segmented = photutils.segmentation.detect_sources(image_co
catalogue = photutils.segmentation.SourceCatalog(data=image_skys
table_source = catalogue.to_table()
```

once the catalogue is created I will store it

```
astropy.io.ascii.write(table_source, "./S13/"+input+".cat", forr
```

Star matching

```
table1 = table.Table.read("./S13/optical.cat",format='ascii.comm
table2 = table.Table.read("./S13/IR.cat",format='ascii.commente

list_source1_x = list (table1['xcentroid'])
list_source1_y = list (table1['ycentroid'])
list_source2_x = list (table2['xcentroid'])
list_source2_y = list (table2['ycentroid'])
position_1= np.transpose ( (list_source1_x, list_source1_y) )
position_2= np.transpose ( (list_source2_x, list_source2_y) )

transf, (list_matched_2, list_matched_1) = astroalign.find_trans
```

This code is taking each catalogue file and try to make star match together, then he will try to make triangle of star to know how to rotate the image.

It’s where my code crash so I cant assure the part after are working since I cannot test them. I think the problem come from the Blur you can see on the image. I you have a feedback on this I would like to hear where the error is coming from and how to correct it.