Advanced Astronomical Observations 2022
Session 12: Aperture Photometry of Real Stars

Kinoshita Daisuke

22 April 2022
publicly accessible version

About this file...

- Important information about this file
  - The author of this file is Kinoshita Daisuke.
  - The original version of this file was used for the course "Advanced Astronomical Observations" (course ID: AS6005) offered at Institute of Astronomy, National Central University from February 2022 to June 2022.
  - The file is provided in the hope that it will be useful, but there is no guarantee for the correctness. Use this file at your own risk.
  - If you are willing to use this file for your study, please feel free to use. I'll be very happy to receive feedback from you.
  - If you are willing to use this file for your teaching, please contact to Kinoshita Daisuke. When you use this file partly or entirely, please mention clearly that the author of the original version is Kinoshita Daisuke. Please help me to improve the contents of this file by sending your feedback.
  - Contact address: `https://www.instagram.com/daisuke23888/`

For this session, we try aperture photometry for real stars.

1 Photometry of stars on SDSS image

1.1 Downloading photometric standard star catalogue

Visit following web page and download photometric standard star catalogue.

- u'g'r'i'z' Standard Star Home Page
  - `https://www-star.fnal.gov/` (Fig. 1)
- Extended Northern and Equatorial u'g'r'i'z' Standards
  - `https://www-star.fnal.gov/NorthEqExtension_ugriz/` (Fig. 2)

Try following command to download files.

```
% curl -k -o stds.tar.gz \
? https://www-star.fnal.gov/NorthEqExtension_ugriz/Data/usno40stds.clean.v3.tar.gz
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 65774  100 65774    0     0  27487      0  0:00:02  0:00:02 --:--:-- 27497
% ls -lF
total 1
-rw-r--r--  1 daisuke  taiwan  65774 Apr 21 13:20 stds.tar.gz
```

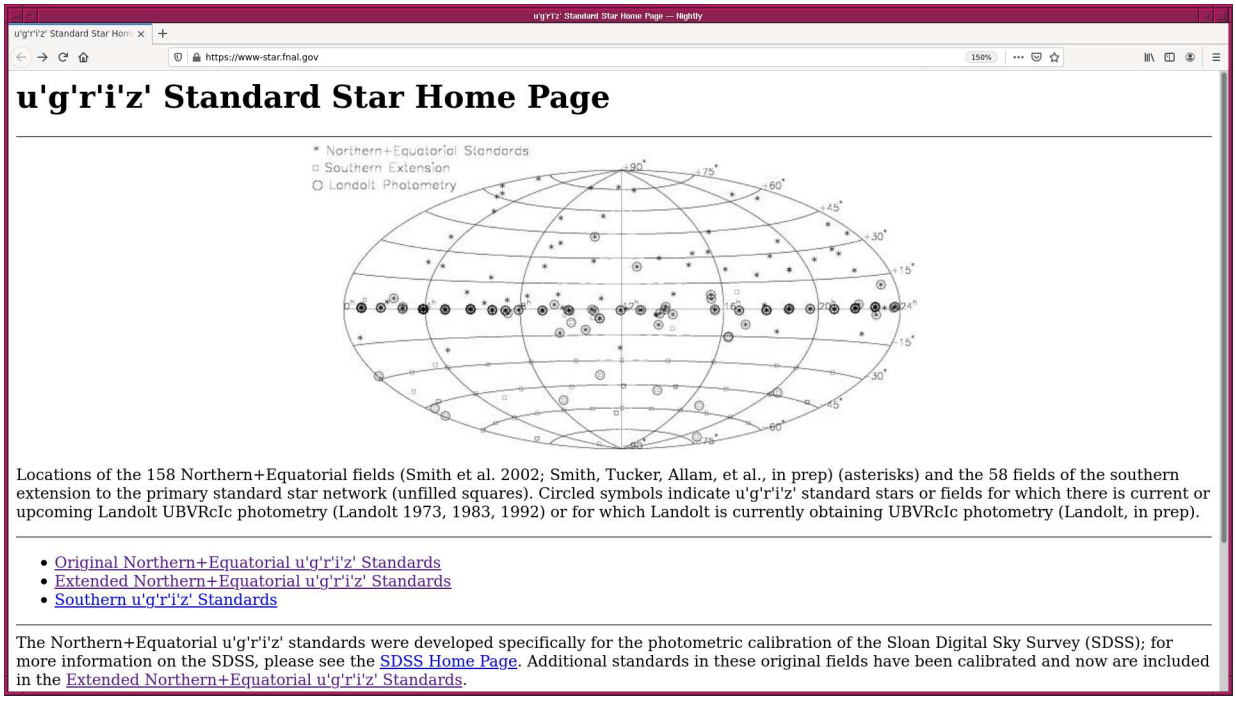

Figure 1: The u'g'r'i'z' Standard Star Home Page.

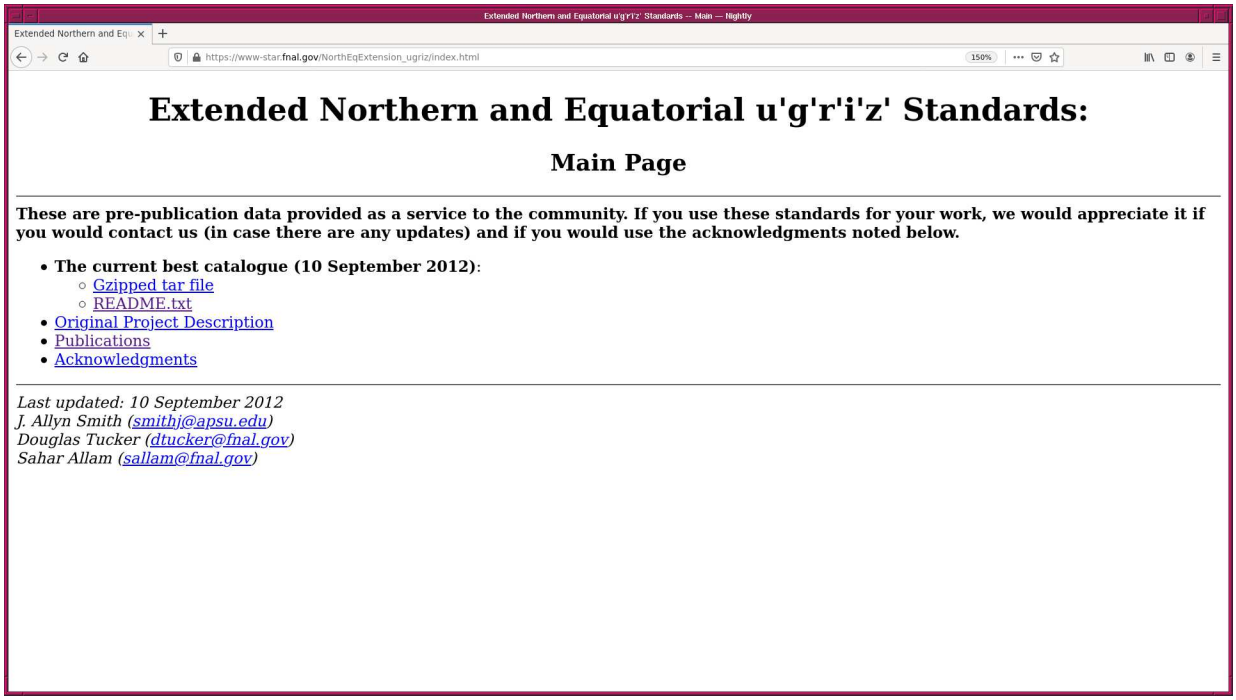

Figure 2: Extended Northern and Equatorial u'g'r'i'z' Standards web page.

Try following command to extract files.

```
% mkdir stds
% ls -lF
total 1
drwxr-xr-x  2 daisuke  taiwan    512 Apr 21 13:21 stds/
-rw-r--r--  1 daisuke  taiwan  65774 Apr 21 13:20 stds.tar.gz
% cd stds
% tar xzvf ../stds.tar.gz
x 100_a.txt.clean.v3
x 100_b.txt.clean.v3
x 101_a.txt.clean.v3
x 101_c.txt.clean.v3
x 104_a.txt.clean.v3
x 105_a.txt.clean.v3
x 107_a.txt.clean.v3
x 107_b.txt.clean.v3
x 108_a.txt.clean.v3
x 108_b.txt.clean.v3

.....

x Ross374.txt.clean.v3
x Ross49.txt.clean.v3
x Ross530.txt.clean.v3
x Ross683.txt.clean.v3
x Ross711.txt.clean.v3
x Ross838.txt.clean.v3
x Ru149.txt.clean.v3
x Wolf1346.txt.clean.v3
x Wolf1447.txt.clean.v3
x Wolf365.txt.clean.v3
% ls
100_a.txt.clean.v3        97_a.txt.clean.v3         GCRV5951.txt.clean.v3
100_b.txt.clean.v3        97_b.txt.clean.v3         GCRV7017.txt.clean.v3
101_a.txt.clean.v3        97_c.txt.clean.v3         GCRV7951.txt.clean.v3
101_c.txt.clean.v3        97_d.txt.clean.v3         GCRV8758.txt.clean.v3
104_a.txt.clean.v3        98_a.txt.clean.v3         GCRV9438.txt.clean.v3

.....

95_b.txt.clean.v3         G15-24.txt.clean.v3       Ru149.txt.clean.v3
95_f.txt.clean.v3         G163_50-51.txt.clean.v3   Wolf1346.txt.clean.v3
96_a.txt.clean.v3         G27-45.txt.clean.v3       Wolf1447.txt.clean.v3
96_b.txt.clean.v3         G3-33.txt.clean.v3        Wolf365.txt.clean.v3
96_c.txt.clean.v3         GCRV5757.txt.clean.v3
% cd ..
% ls -lF
total 1
drwxr-xr-x  2 daisuke  taiwan   3584 Apr 21 13:22 stds/
-rw-r--r--  1 daisuke  taiwan  65774 Apr 21 13:20 stds.tar.gz
```

1.2 Coordinates of photometric standard stars

We process the image of standard star field PG0918+029. Make a Python script to read coordinates of stars in the field around the star PG0918+029.

Python Code 1: advobs202202_s12_01.py

```
#!/usr/pkg/bin/python3.9

#
# Time-stamp: <2022/04/21 14:25:23 (CST) daisuke>
#

# importing argparse module
import argparse

# importing astropy module
import astropy.coordinates

# constructing parser object
desc   = "reading standard star information from file"
parser = argparse.ArgumentParser (description=desc)

# adding arguments
parser.add_argument ('files', nargs='+', help='files to read')

# command-line argument analysis
args = parser.parse_args ()

# list of data files
list_files = args.files

# printing header
print ("# star ID, RA, Dec, r'-band mag.")

for file_data in list_files:
    # opening the file
    with open (file_data, 'r') as fh:
        # reading the file line-by-line
        for line in fh:
            # if the line starts with '#', then skip
            if (line[0] == '#'):
                continue
            # splitting the data
            records = line.split ()
            # star ID
            star_id = int (records[0])
            # RA
            ra_deg  = float (records[1])
            # Dec
            dec_deg = float (records[2])
            # r'-band magnitude
            mag_r   = float (records[9])
            # coordinates
            coord = astropy.coordinates.SkyCoord (ra_deg, dec_deg, unit='deg')
            # conversion into hmsdms format
            radec_str = coord.to_string ('hmsdms')
            ra_str    = radec_str.split ()[0]
            dec_str   = radec_str.split ()[1]
            # printing coordinate
            print ("%03d  %-16s  %-16s  %6.3f" \
                   % (star_id, ra_str, dec_str, mag_r) )
```

Execute the script, and show the coordinates of stars.

```
% chmod a+x advobs202202_s12_01.py
% ./advobs202202_s12_01.py -h
usage: advobs202202_s12_01.py [-h] files [files ...]

reading standard star information from file

positional arguments:
  files       files to read

optional arguments:
  -h, --help  show this help message and exit

% ./advobs202202_s12_01.py stds/PG0918+029D.txt.clean.v3
# star ID, RA, Dec, r'-band mag.
005  09h21m24.3288s    +02d47m54.6s      11.720
006  09h21m21.9432s    +02d47m28.32s     11.939
007  09h21m47.1456s    +02d49m32.16s     12.427
008  09h21m19.0584s    +02d50m37.68s     13.164
009  09h21m42.312s     +02d46m37.2s      13.374
010  09h21m28.224s     +02d46m02.28s     13.559
011  09h21m32.9328s    +02d47m59.28s     13.753
012  09h21m35.1168s    +02d46m19.56s     14.339
013  09h21m16.7136s    +02d42m19.8s      14.477
014  09h21m40.8792s    +02d51m21.24s     14.611
016  09h21m32.4936s    +02d51m09.72s     14.839
```

1.3 Downloading SDSS image

Make a Python script to download SDSS image around the star PG0918+029.

Python Code 2: advobs202202_s12_02.py

```
#!/usr/pkg/bin/python3.9

#
# Time-stamp: <2022/04/21 20:59:22 (CST) daisuke>
#

# importing argparse module
import argparse

# importing sys module
import sys

# importing pathlib module
import pathlib

# importing astroquery module
import astroquery.simbad
import astroquery.ipac.ned
import astroquery.skyview

# importing astropy module
import astropy.coordinates
import astropy.units

# importing datetime module
import datetime
```

```
# importing ssl module
import ssl

# allow insecure downloading
ssl._create_default_https_context = ssl._create_unverified_context

# date/time
now = datetime.datetime.now ().isoformat ()

# units
u_ha  = astropy.units.hourangle
u_deg = astropy.units.deg

# constructing parser object
desc   = "downloading SDSS image"
parser = argparse.ArgumentParser (description=desc)

# adding arguments
list_resolver = ['simbad', 'ned']
list_survey   = ['DSS1 Blue', 'DSS1 Red', 'DSS2 Blue', 'DSS2 Red', 'DSS2 IR', \
                 'SDSSu', 'SDSSg', 'SDSSr', 'SDSSi', 'SDSSz', \
                 'SDSSdr7u', 'SDSSdr7g', 'SDSSdr7r', 'SDSSdr7i', 'SDSSdr7z']
parser.add_argument ('-r', '--resolver', choices=list_resolver, \
                     default='simbad', help='choice of name resolver')
parser.add_argument ('-s', '--survey', choices=list_survey, \
                     default='SDSSr', help='choice of survey')
parser.add_argument ('-t', '--target', default='', help='target name')
parser.add_argument ('-f', '--fov', type=int, default=1024, \
                     help='field-of-view in pixel')
parser.add_argument ('-o', '--output', default='', help='output file name')

# command-line argument analysis
args = parser.parse_args ()

# input parameters
name_resolver = args.resolver
survey        = args.survey
target_name   = args.target
fov_pix       = args.fov
file_output   = args.output

# checking target name
if (target_name == ''):
    # printing error message
    print ("No target name is given!")
    # exit
    sys.exit ()

# making pathlib object
path_output = pathlib.Path (file_output)

# checking output file name
if (file_output == ''):
    # printing error message
    print ("No output file name is given!")
    # exit
    sys.exit ()
elif not (path_output.suffix == '.fits'):
```

```
    # printing error message
    print ("Output file must be FITS file!")
    # exit
    sys.exit ()
elif (path_output.exists ()):
    # printing error message
    print ("Output file exists!")
    # exit
    sys.exit ()

# using name resolver
if (name_resolver == 'simbad'):
    query_result = astroquery.simbad.Simbad.query_object (target_name)
elif (name_resolver == 'ned'):
    query_result = astroquery.ipac.ned.Ned.query_object (target_name)

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

# coordinate
if (name_resolver == 'simbad'):
    coord = astropy.coordinates.SkyCoord (RA[0], Dec[0], unit=(u_ha, u_deg))
elif (name_resolver == 'ned'):
    coord = astropy.coordinates.SkyCoord (RA[0], Dec[0], unit=(u_deg, u_deg))

coord_str = coord.to_string (style='hmsdms')
(coord_ra_str, coord_dec_str) = coord_str.split ()
coord_ra_deg  = coord.ra.deg
coord_dec_deg = coord.dec.deg

# printing coordinate
print ("Target Name: %s" % target_name)
print ("  RA:  %s = %f deg" % (coord_ra_str, coord_ra_deg) )
print ("  Dec: %s = %f deg" % (coord_dec_str, coord_dec_deg) )

# searching image
list_image = astroquery.skyview.SkyView.get_image_list (position=coord, \
                                                         survey=survey)

# printing image list
print ("Available images:")
print (" ", list_image)

# getting image
image = astroquery.skyview.SkyView.get_images (position=coord, \
                                               survey=survey, pixels=fov_pix)

# header and data
image0 = image[0]
header = image0[0].header
data   = image0[0].data

# adding comments in header
header['history'] = "image downloaded from %s" % survey
header['history'] = "image saved on %s" % now

# saving to a FITS file
astropy.io.fits.writeto (file_output, data, header=header)
```

Run the script, and download the image.

```
% chmod a+x advobs202202_s12_02.py
% ./advobs202202_s12_02.py -h
usage: advobs202202_s12_02.py [-h] [-r {simbad,ned}]
                              [-s {SDSSu,SDSSg,SDSSr,SDSSi,SDSSz,SDSSdr7u,SDSSdr
7g,SDSSdr7r,SDSSdr7i,SDSSdr7z}]
                              [-t TARGET] [-f FOV] [-o OUTPUT]

downloading SDSS image

optional arguments:
  -h, --help            show this help message and exit
  -r {simbad,ned}, --resolver {simbad,ned}
                        choice of name resolver
  -s {SDSSu,SDSSg,SDSSr,SDSSi,SDSSz,SDSSdr7u,SDSSdr7g,SDSSdr7r,SDSSdr7i,SDSSdr7z
}, --survey {SDSSu,SDSSg,SDSSr,SDSSi,SDSSz,SDSSdr7u,SDSSdr7g,SDSSdr7r,SDSSdr7i,S
DSSdr7z}
                        choice of survey
  -t TARGET, --target TARGET
                        target name
  -f FOV, --fov FOV     field-of-view in pixel
  -o OUTPUT, --output OUTPUT
                        output file name

% ./advobs202202_s12_02.py -r simbad -s SDSSr -t PG0918+029 -f 2048 \
? -o pg0918_sdss_r.fits
Target Name: PG0918+029
  RA:  09h21m28.2138s = 140.367557 deg
  Dec: +02d46m02.254s = 2.767293 deg
Available images:
  ['https://skyview.gsfc.nasa.gov/tempspace/fits/skv4783212841137.fits']
Downloading https://skyview.gsfc.nasa.gov/tempspace/fits/skv4783217584191.fits
|=========================================|  16M/ 16M (100.00%)        36s

% ls -lF *.fits
-rw-r--r--  1 daisuke  taiwan  16793280 Apr 21 14:32 pg0918_sdss_r.fits
```

1.4 Visualisation of SDSS image

Use Ginga to visualise the SDSS image. (Fig. 3)

```
% ginga pg0918_sdss_r.fits
```

1.5 Marking standard stars

Make a Python script to show the positions of photometric standard stars.

Python Code 3: advobs202202_s12_03.py

```
#!/usr/pkg/bin/python3.9

#
# Time-stamp: <2022/04/21 15:59:32 (CST) daisuke>
#
```

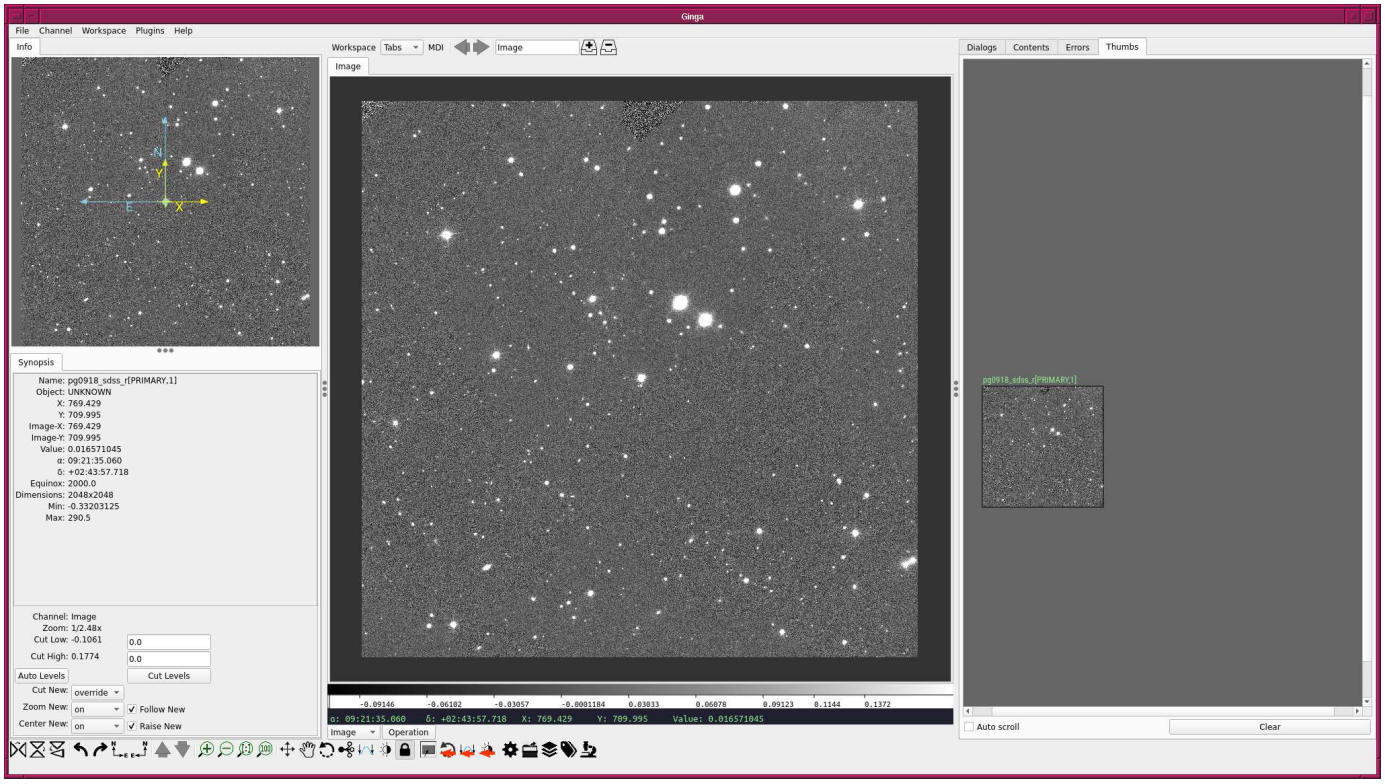

Figure 3: SDSS image around the star PG0918+029.

```
# importing argparse module
import argparse

# importing sys module
import sys

# importing pathlib module
import pathlib

# importing numpy module
import numpy

# importing astropy module
import astropy.io.fits
import astropy.units
import astropy.wcs

# importing photutils module
import photutils.aperture

# importing matplotlib module
import matplotlib.figure
import matplotlib.backends.backend_agg
import matplotlib.patches

# constructing parser object
desc   = 'marking standard stars'
parser = argparse.ArgumentParser (description=desc)

# colour maps
```

```
choices_cmap = ['viridis', 'plasma', 'inferno', 'magma', 'cividis', \
                'binary', 'gray', 'bone', 'pink', \
                'spring', 'summer', 'autumn', 'winter', \
                'cool', 'hot', 'copper', 'ocean', 'terrain', \
                'gnuplot', 'cubehelix', 'jet', 'turbo']

# adding command-line arguments
parser.add_argument ('-c', '--cmap', default='bone', choices=choices_cmap, \
                     help='choice of colour map (default: bone)')
parser.add_argument ('-d', '--resolution', type=int, default=450, \
                     help='resolution in DPI (default: 450)')
parser.add_argument ('-r', '--radius', type=float, default=20.0, \
                     help='radius of aperture circle in arcsec')
parser.add_argument ('-w', '--width', type=float, default=2.0, \
                     help='width of circle')
parser.add_argument ('-s', '--std', default='', help='standard star file')
parser.add_argument ('-o', '--output', default='', \
                     help='output image file name')
parser.add_argument ('file', default='', help='input FITS file name')

# command-line argument analysis
args = parser.parse_args ()

# input parameters
cmap          = args.cmap
resolution    = args.resolution
radius_arcsec = args.radius
width         = args.width
file_std      = args.std
file_output   = args.output
file_fits     = args.file

# making pathlib objects
path_fits   = pathlib.Path (file_fits)
path_std    = pathlib.Path (file_std)
path_output = pathlib.Path (file_output)

# checking input FITS file
if (file_fits == ''):
    # printing message
    print ("You need to specify input file name.")
    # exit
    sys.exit ()
if not (path_fits.suffix == '.fits'):
    # printing message
    print ("Input file must be a FITS file.")
    # exit
    sys.exit ()
if not (path_fits.suffix == '.fits'):
    # printing message
    print ("Input file must be a FITS file.")
    # exit
    sys.exit ()

# checking standard star
if (file_std == ''):
    # printing message
    print ("You need to specify input file name.")
    # exit
```

```
    sys.exit ()
if not (path_std.exists ()):
    # printing message
    print ("Input standard star file '%s' does not exist." % file_std)
    # exit
    sys.exit ()

# checking output image file
if not ( (path_output.suffix == '.eps') or (path_output.suffix == '.pdf') \
         or (path_output.suffix == '.png') or (path_output.suffix == '.ps') ):
    # printing message
    print ("Output file must be either EPS, PDF, PNG, or PS.")
    # exit
    sys.exit ()
if (path_output.exists ()):
    # printing message
    print ("Output image file '%s' exists." % file_output)
    # exit
    sys.exit ()

# opening FITS file
with astropy.io.fits.open (file_fits) as hdu_list:
    # reading FITS header
    header = hdu_list[0].header

    # reading WCS information from header
    wcs = astropy.wcs.WCS (header)

    # reading FITS image data
    data = hdu_list[0].data

# making empty list
list_coords = []

# opening standard star file
with open (file_std, 'r') as fh:
    # reading file line-by-line
    for line in fh:
        # skip if the line stars with '#'
        if (line[0] == '#'):
            continue
        # splitting data
        records = line.split ()
        # RA
        ra_deg = float (records[1])
        # Dec
        dec_deg = float (records[2])
        # RA and Dec
        coord = astropy.coordinates.SkyCoord (ra_deg, dec_deg, unit='deg')
        # appending coord to list_coords
        list_coords.append (coord)

# radius of circle in pixel
pix_scale    = abs (header['CDELT1']) * 3600.0
radius_pixel = radius_arcsec / pix_scale

# making objects "fig" and "ax"
fig = matplotlib.figure.Figure ()
matplotlib.backends.backend_agg.FigureCanvasAgg (fig)
```

```
ax = fig.add_subplot (111, projection=wcs)

# axes
ax.set_xlabel ('RA')
ax.set_ylabel ('Dec')

# plotting image
norm \
    = astropy.visualization.mpl_normalize.ImageNormalize \
    ( stretch=astropy.visualization.HistEqStretch (data) )
im = ax.imshow (data, origin='lower', cmap=cmap, norm=norm)
fig.colorbar (im)

# plotting locations of standard stars
for coord in list_coords:
    # conversion from world coordinate (RA, Dec) into pixel coordinate (x, y)
    (x, y) = wcs.world_to_pixel (coord)
    # making a circle
    stdstars = matplotlib.patches.Circle (xy=(x, y), \
                                          radius=radius_pixel, \
                                          fill=False, color="red", \
                                          linewidth=width)
    # plotting location of standard star
    ax.add_patch (stdstars)

# saving file
fig.savefig (file_output, dpi=resolution)
```

Execute the script, and show the positions of photometric starndard stars.

```
% chmod a+x advobs202202_s12_03.py
% ./advobs202202_s12_03.py -h
usage: advobs202202_s12_03.py [-h]
                              [-c {viridis,plasma,inferno,magma,cividis,binary,g
ray,bone,pink,spring,summer,autumn,winter,cool,hot,copper,ocean,terrain,gnuplot,
cubehelix,jet,turbo}]
                              [-d RESOLUTION] [-r RADIUS] [-w WIDTH] [-s STD]
                              [-o OUTPUT]
                              file

marking standard stars

positional arguments:
  file                  input FITS file name

optional arguments:
  -h, --help            show this help message and exit
  -c {viridis,plasma,inferno,magma,cividis,binary,gray,bone,pink,spring,summer,a
utumn,winter,cool,hot,copper,ocean,terrain,gnuplot,cubehelix,jet,turbo}, --cmap
{viridis,plasma,inferno,magma,cividis,binary,gray,bone,pink,spring,summer,autumn
,winter,cool,hot,copper,ocean,terrain,gnuplot,cubehelix,jet,turbo}
                        choice of colour map (default: bone)
  -d RESOLUTION, --resolution RESOLUTION
                        resolution in DPI (default: 450)
  -r RADIUS, --radius RADIUS
                        radius of aperture circle in arcsec
  -w WIDTH, --width WIDTH
                        width of circle
```

```
  -s STD, --std STD     standard star file
  -o OUTPUT, --output OUTPUT
                        output image file name

% ./advobs202202_s12_03.py -s stds/PG0918+029D.txt.clean.v3 -c viridis \
? -o pg0918_std.png pg0918_sdss_r.fits
% ls -lF *.png
-rw-r--r--  1 daisuke  taiwan  8513543 Apr 21 16:03 pg0918_std.png
```

Show the image file created by the script. (Fig. 4)

```
% feh -dF pg0918_std.png
```

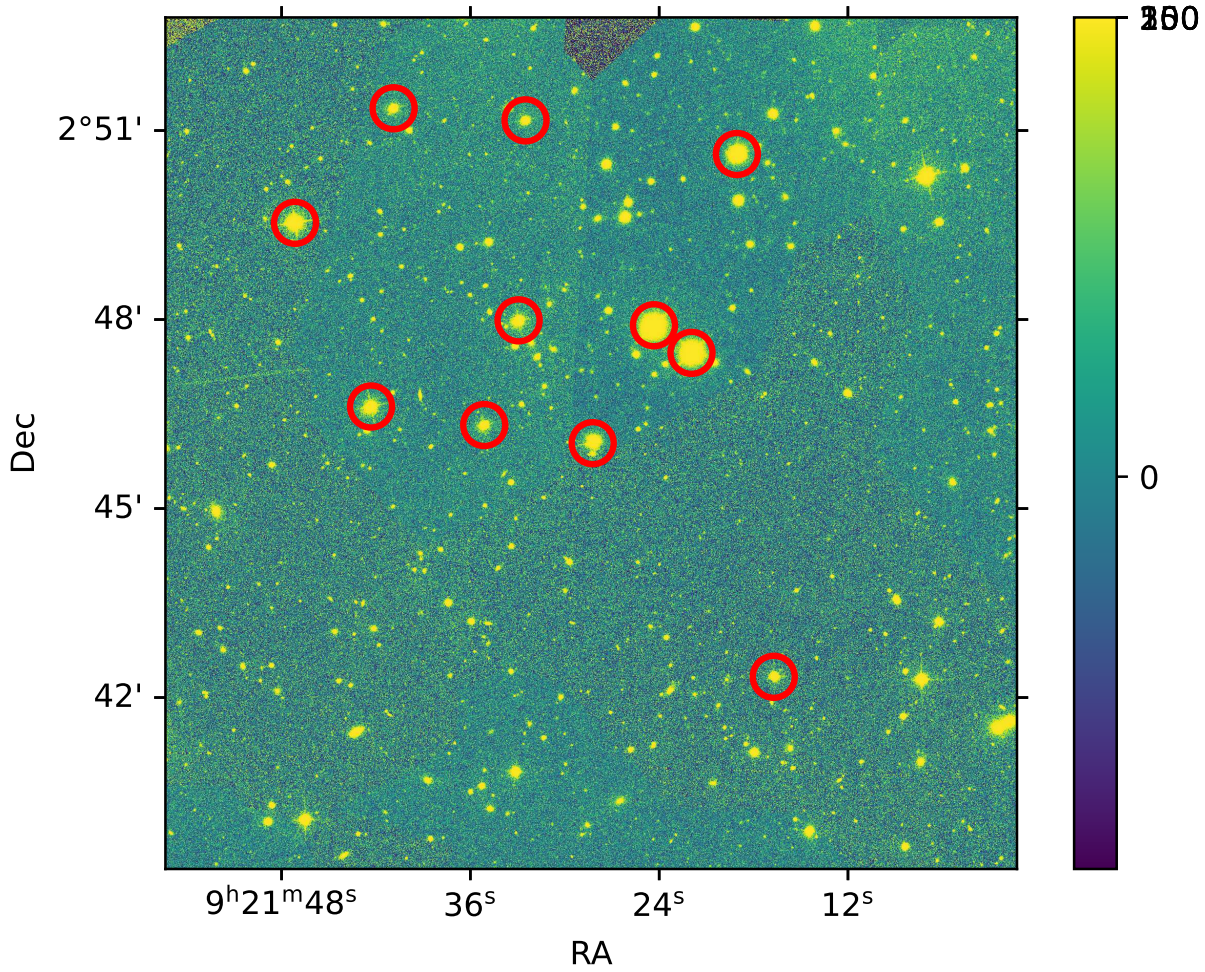

Figure 4: Locations of photometric standard stars in PG0918+029 field.

1.6 Choosing two stars

Pick two stars for your measurements. Choose two stars on the image `pg0918_std.png`. Then, use Ginga to find rough values of RA and Dec of those two stars. Then, find those two stars on the standard star file.

For example, following two stars are selected. (Fig. 5 and 6) These are stars of ID 12 and 16.

```
012   09h21m35.1168s     +02d46m19.56s      14.339
016   09h21m32.4936s     +02d51m09.72s      14.839
```

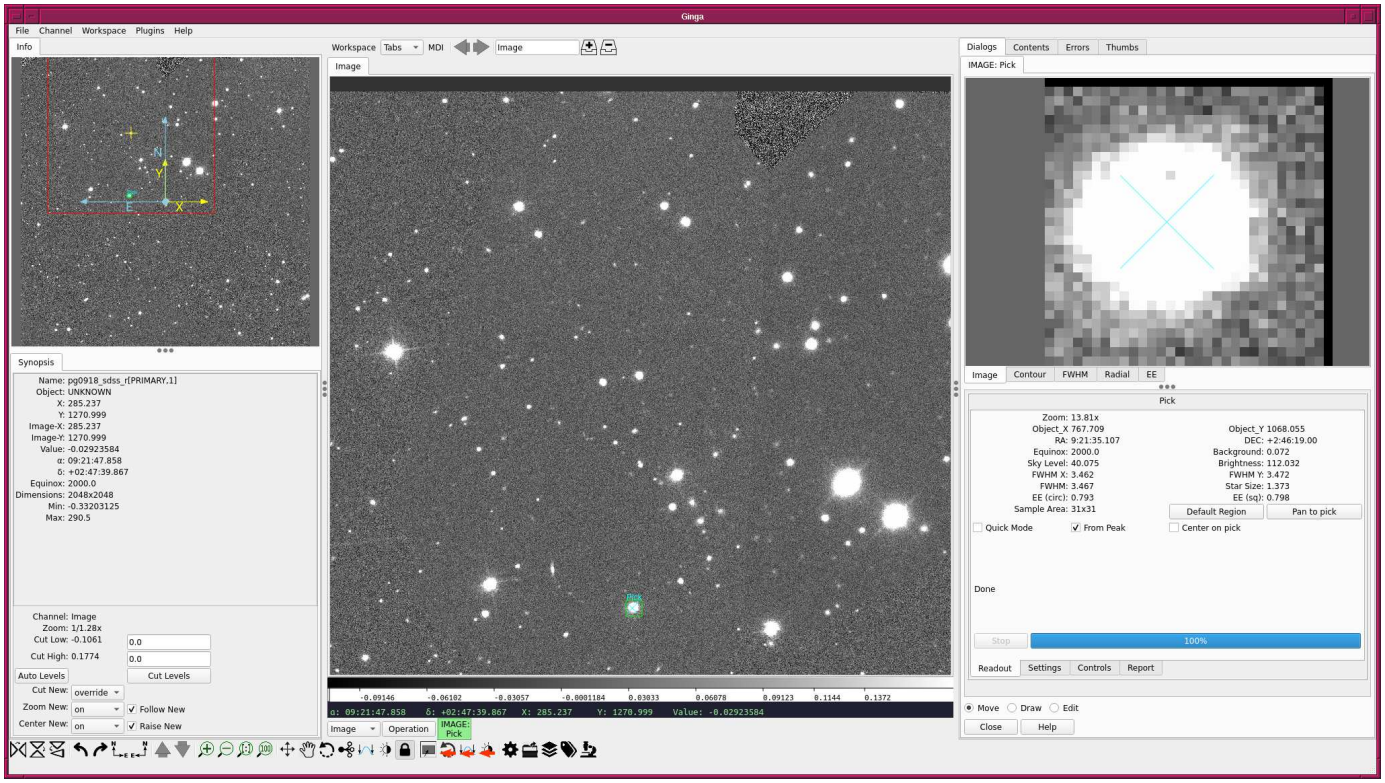

Figure 5: First star for the measurement.

1.7 Measuring accurate positions of stars using centroid

Make a Python script to measure the position and FWHM of PSF using 2-dimensional Gaussian function.

Python Code 4: advobs202202_s12_04.py

```
#!/usr/pkg/bin/python3.9

#
# Time-stamp: <2022/04/14 12:49:58 (CST) daisuke>
#

# importing argparse module
import argparse

# importing sys module
import sys

# importing pathlib module
import pathlib

# importing numpy module
import numpy

# importing astropy module
import astropy.io.fits
```

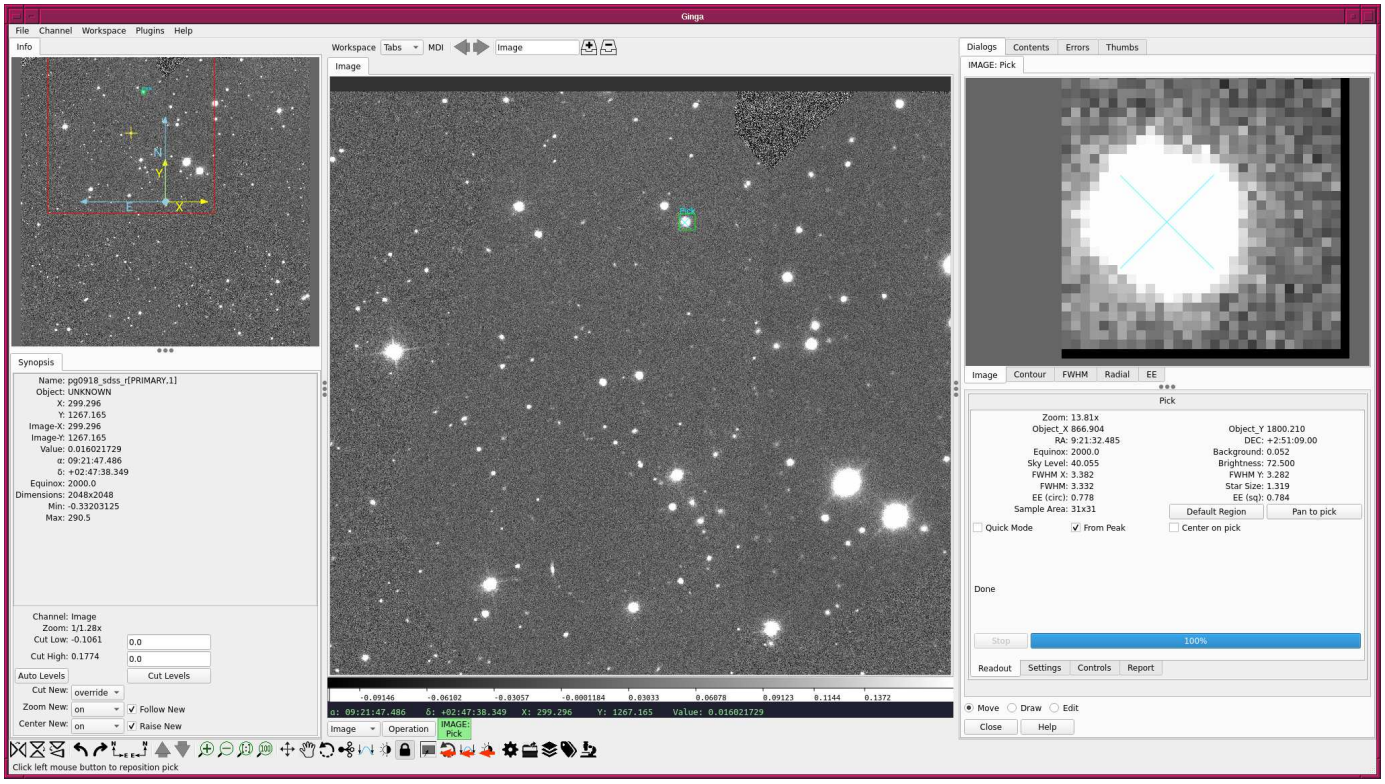

Figure 6: Second star for the measurement.

```
import astropy.modeling

# importing photutils module
import photutils.centroids

# importing matplotlib module
import matplotlib.figure
import matplotlib.backends.backend_agg
import matplotlib.patches

# constructing parser object
desc   = 'PSF fitting for a point-source object'
parser = argparse.ArgumentParser (description=desc)

# centroid measurement technique
choices_centroid = ['com', '1dg', '2dg']

# PSF models (Gaussian and Moffat)
choices_psf = ['2dg', '2dm']

# colour maps
choices_cmap = ['viridis', 'plasma', 'inferno', 'magma', 'cividis', \
                'binary', 'gray', 'bone', 'pink', \
                'spring', 'summer', 'autumn', 'winter', \
                'cool', 'hot', 'copper', 'ocean', 'terrain', \
                'gnuplot', 'cubehelix', 'jet', 'turbo']

# adding command-line arguments
parser.add_argument ('-c', '--centroid', choices=choices_centroid, \
                     default='com', \
```

```
                      help='centroid measurement algorithm (default: com)')
parser.add_argument ('-p', '--psf', choices=choices_psf, default='2dg', \
                      help='PSF model [2dg=Gaussian, 2dm=Moffat] (default: 2dg)')
parser.add_argument ('-w', '--width', type=int, default=5, \
                      help='half-width of centroid calculation box (default: 5)')
parser.add_argument ('-x', '--xinit', type=int, default=-1, \
                      help='a rough x coordinate of target')
parser.add_argument ('-y', '--yinit', type=int, default=-1, \
                      help='a rough y coordinate of target')
parser.add_argument ('-r', '--resolution', type=int, default=450, \
                      help='resolution in DPI (default: 450)')
parser.add_argument ('-m', '--cmap', default='bone', choices=choices_cmap, \
                      help='choice of colour map (default: bone)')
parser.add_argument ('-o', '--output', default='centroid.png', \
                      help='output file name (default: centroid.png)')
parser.add_argument ('file', default='', help='input file name')

# command-line argument analysis
args = parser.parse_args ()

# input parameters
half_width  = args.width
x_init      = args.xinit
y_init      = args.yinit
centroid    = args.centroid
psf_model   = args.psf
resolution  = args.resolution
cmap        = args.cmap
file_fits   = args.file
file_output = args.output

# making pathlib objects
path_fits   = pathlib.Path (file_fits)
path_output = pathlib.Path (file_output)

# checking input FITS file name
if (file_fits == ''):
    # printing message
    print ("You need to specify input file name.")
    # exit
    sys.exit ()
# if input file is not a FITS file, then stop the script
if not (path_fits.suffix == '.fits'):
    # printing message
    print ("Input file must be a FITS file.")
    # exit
    sys.exit ()
# if input file does not exist, then stop the script
if not (path_fits.exists ()):
    # printing message
    print ("Input file does not exist.")
    # exit
    sys.exit ()
# if output file exists, then stop the script
if (path_output.exists ()):
    # printing message
    print ("Output file exists.")
    # exit
    sys.exit ()
```

```
# output file must be either EPS, PDF, PNG, or PS file
if not ( (path_output.suffix == '.eps') or (path_output.suffix == '.pdf') \
         or (path_output.suffix == '.png') or (path_output.suffix == '.ps') ):
    # printing message
    print ("Output file must be either EPS, PDF, PNG, or PS file.")
    # exit
    sys.exit ()

# printing information
print ("#")
print ("# input parameters")
print ("#")
print ("#  input file name          = %s" % file_fits)
print ("#  half-width of search box = %f" % half_width)
print ("#  x_init                   = %f" % x_init)
print ("#  y_init                   = %f" % y_init)
print ("#  centroid technique       = %s" % centroid)
print ("#  output file name         = %s" % file_output)
print ("#")

# printing status
print ("# now, reading FITS file...")

# opening FITS file
with astropy.io.fits.open (file_fits) as hdu_list:
    # reading FITS header
    header = hdu_list[0].header

    # image size
    image_size_x = header['NAXIS1']
    image_size_y = header['NAXIS2']

    # checking x_init and y_init
    if not ( (x_init > 0) and (x_init < image_size_x) ):
        print ("Input x_init value exceed image size.")
        sys.exit ()
    if not ( (y_init > 0) and (y_init < image_size_y) ):
        print ("Input y_init value exceed image size.")
        sys.exit ()

    # reading FITS image data
    data = hdu_list[0].data

# printing status
print ("# finished reading FITS file!")

# printing status
print ("# now, extracting image around the target object...")

# region of calculation
x_min = x_init - half_width
x_max = x_init + half_width + 1
y_min = y_init - half_width
y_max = y_init + half_width + 1

# extraction of subframe for calculation
subframe = data[y_min:y_max, x_min:x_max]

# printing status
```

```
print ("# finished extracting image around the target object!")

# printing status
print ("# now, subtracting background...")

# rough background subtraction
subframe -= numpy.median (subframe)

# printing status
print ("# finished subtracting background!")

# printing status
print ("# now, measuring centroid...")

# centroid calculation
if (centroid == 'com'):
    (x_centre, y_centre) = photutils.centroids.centroid_com (subframe)
elif (centroid == '1dg'):
    (x_centre, y_centre) = photutils.centroids.centroid_1dg (subframe)
elif (centroid == '2dg'):
    (x_centre, y_centre) = photutils.centroids.centroid_2dg (subframe)

# printing status
print ("# finished measuring!")

# printing status
print ("# now, measuring PSF...")

# PSF fitting
subframe_y, subframe_x = numpy.indices (subframe.shape)
if (psf_model == '2dg'):
    psf_init = astropy.modeling.models.Gaussian2D (x_mean=x_centre, \
                                                   y_mean=y_centre)
elif (psf_model == '2dm'):
    psf_init = astropy.modeling.models.Moffat2D (x_0=x_centre, y_0=y_centre, \
                                                 amplitude=1.0, \
                                                 alpha=1.0, gamma=1.0)
fit = astropy.modeling.fitting.LevMarLSQFitter ()
psf_fitted = fit (psf_init, subframe_x, subframe_y, subframe)

# result of fitting
amplitude = psf_fitted.amplitude.value
if (psf_model == '2dg'):
    x_centre_sub = psf_fitted.x_mean.value
    y_centre_sub = psf_fitted.y_mean.value
    x_centre_psf = psf_fitted.x_mean.value + x_min
    y_centre_psf = psf_fitted.y_mean.value + y_min
    x_fwhm       = psf_fitted.x_fwhm
    y_fwhm       = psf_fitted.y_fwhm
    fwhm         = (x_fwhm + y_fwhm) / 2.0
    theta        = psf_fitted.theta.value
if (psf_model == '2dm'):
    x_centre_sub = psf_fitted.x_0.value
    y_centre_sub = psf_fitted.y_0.value
    x_centre_psf = psf_fitted.x_0.value + x_min
    y_centre_psf = psf_fitted.y_0.value + y_min
    alpha        = psf_fitted.alpha.value
    gamma        = psf_fitted.gamma.value
    fwhm         = psf_fitted.fwhm
```

```
# printing status
print ("# finished measuring PSF!")

# printing result
print ("#")
print ("# result of the measurement")
print ("#")
print ("#  x_centre  = %f" % x_centre_psf)
print ("#  y_centre  = %f" % y_centre_psf)
print ("#  amplitude = %f" % amplitude)
if (psf_model == '2dg'):
    print ("#  x_fwhm    = %f" % x_fwhm)
    print ("#  y_fwhm    = %f" % y_fwhm)
    print ("#  theta     = %f" % theta)
elif (psf_model == '2dm'):
    print ("#  alpha     = %f" % alpha)
    print ("#  gamma     = %f" % gamma)
    print ("#  fwhm      = %f" % fwhm)
print ("#")
print ("# X_CENTRE, Y_CENTRE, FWHM")
print ("%f %f %f" % (x_centre_psf, y_centre_psf, fwhm) )

# printing status
print ("# now, generating a plot...")

# making objects "fig" and "ax"
fig = matplotlib.figure.Figure ()
matplotlib.backends.backend_agg.FigureCanvasAgg (fig)
ax = fig.add_subplot (111)

# axes
ax.set_xlabel ('X [pixel]')
ax.set_ylabel ('Y [pixel]')

# plotting image
im = ax.imshow (subframe, origin='lower', cmap=cmap)
fig.colorbar (im)
ax.plot (x_centre_sub, y_centre_sub, marker='+', color='red', markersize=10)

# adding a bar to represent FWHM
bar = matplotlib.patches.Rectangle (xy=(1,1), width=fwhm, height=1.0, \
                                    facecolor='green', edgecolor='white')
ax.add_patch (bar)

# saving file
fig.savefig (file_output, dpi=resolution)

# printing status
print ("# finished generating a plot!")
```

Execute the script to measure centroid and FWHM of the star ID 12.

```
% chmod a+x advobs202202_s12_04.py
% ./advobs202202_s12_04.py -h
usage: advobs202202_s12_04.py [-h] [-c {com,1dg,2dg}] [-p {2dg,2dm}]
                              [-w WIDTH] [-x XINIT] [-y YINIT] [-r RESOLUTION]
                              [-m {viridis,plasma,inferno,magma,cividis,binary,g
```

```
ray,bone,pink,spring,summer,autumn,winter,cool,hot,copper,ocean,terrain,gnuplot,
cubehelix,jet,turbo}]
                              [-o OUTPUT]
                              file

PSF fitting for a point-source object

positional arguments:
  file                  input file name

optional arguments:
  -h, --help            show this help message and exit
  -c {com,1dg,2dg}, --centroid {com,1dg,2dg}
                        centroid measurement algorithm (default: com)
  -p {2dg,2dm}, --psf {2dg,2dm}
                        PSF model [2dg=Gaussian, 2dm=Moffat] (default: 2dg)
  -w WIDTH, --width WIDTH
                        half-width of centroid calculation box (default: 5)
  -x XINIT, --xinit XINIT
                        a rough x coordinate of target
  -y YINIT, --yinit YINIT
                        a rough y coordinate of target
  -r RESOLUTION, --resolution RESOLUTION
                        resolution in DPI (default: 450)
  -m {viridis,plasma,inferno,magma,cividis,binary,gray,bone,pink,spring,summer,a
utumn,winter,cool,hot,copper,ocean,terrain,gnuplot,cubehelix,jet,turbo}, --cmap
{viridis,plasma,inferno,magma,cividis,binary,gray,bone,pink,spring,summer,autumn
,winter,cool,hot,copper,ocean,terrain,gnuplot,cubehelix,jet,turbo}
                        choice of colour map (default: bone)
  -o OUTPUT, --output OUTPUT
                        output file name (default: centroid.png)

% ./advobs202202_s12_04.py -c 2dg -p 2dg -w 20 -x 768 -y 1068 -m viridis \
? -o star01_centroid.png pg0918_sdss_r.fits
#
# input parameters
#
#  input file name          = pg0918_sdss_r.fits
#  half-width of search box = 20.000000
#  x_init                   = 768.000000
#  y_init                   = 1068.000000
#  centroid technique       = 2dg
#  output file name         = star01_centroid.png
#
# now, reading FITS file...
# finished reading FITS file!
# now, extracting image around the target object...
# finished extracting image around the target object!
# now, subtracting background...
# finished subtracting background!
# now, measuring centroid...
# finished measuring!
# now, measuring PSF...
# finished measuring PSF!
#
# result of the measurement
#
#  x_centre  = 766.726015
#  y_centre  = 1067.012440
```

```
#  amplitude = 112.482397
#  x_fwhm    = 3.846640
#  y_fwhm    = 3.329685
#  theta     = 0.766968
#
# X_CENTRE, Y_CENTRE, FWHM
766.726015 1067.012440 3.588162
# now, generating a plot...
# finished generating a plot!
```

The star ID 12 is located at $(x, y) = (766.73, 1067.01)$ and its FWHM is 3.59 pixel. Check the image. (7)

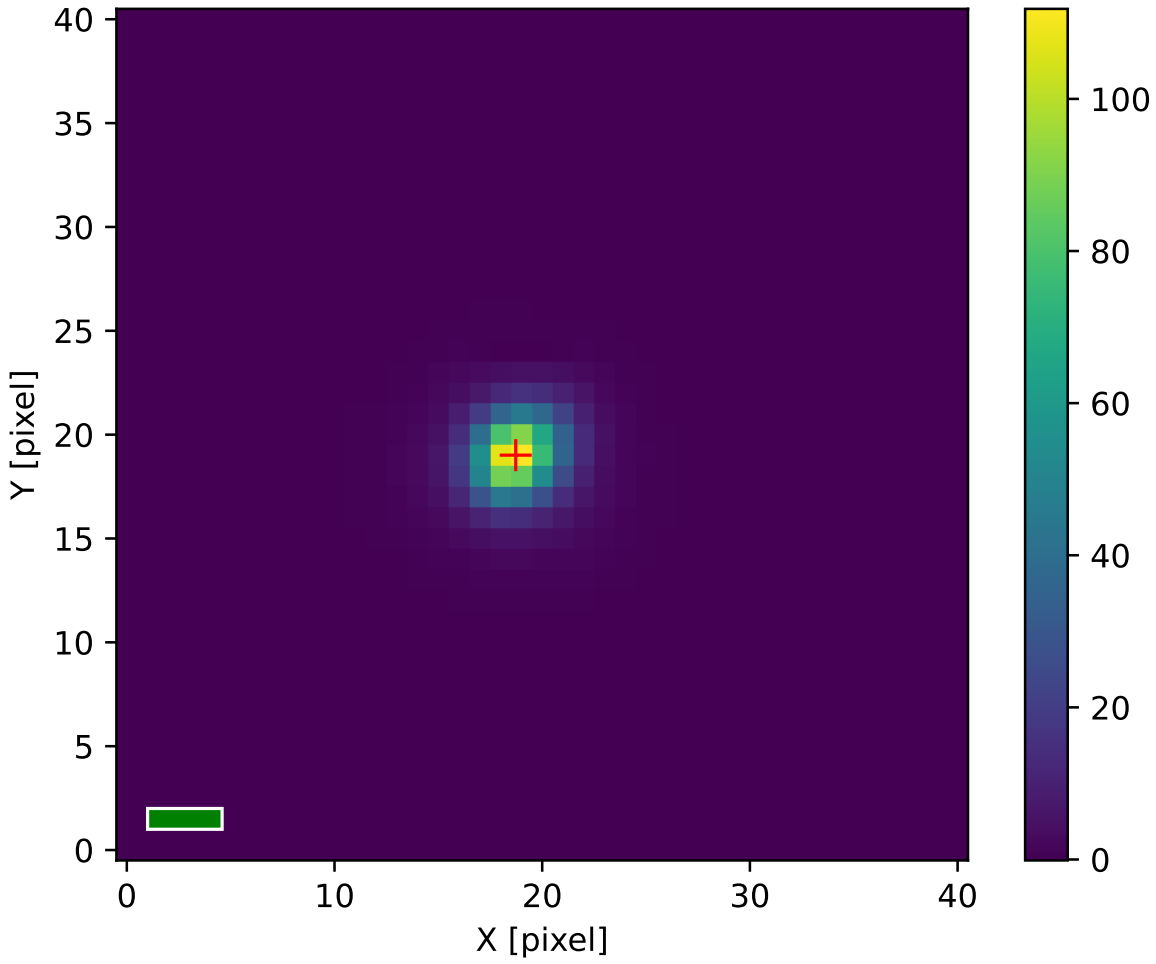

Figure 7: The results of centroid and FWHM measurement for the star ID 12.

Next, measure centroid and FWHM of the star ID 16.

```
% ./advobs202202_s12_04.py -c 2dg -p 2dg -w 20 -x 867 -y 1800 -m viridis \
? -o star02_centroid.png pg0918_sdss_r.fits
#
# input parameters
#
#  input file name          = pg0918_sdss_r.fits
```

```
#  half-width of search box = 20.000000
#  x_init                   = 867.000000
#  y_init                   = 1800.000000
#  centroid technique       = 2dg
#  output file name         = star02_centroid.png
#
# now, reading FITS file...
# finished reading FITS file!
# now, extracting image around the target object...
# finished extracting image around the target object!
# now, subtracting background...
# finished subtracting background!
# now, measuring centroid...
# finished measuring!
# now, measuring PSF...
# finished measuring PSF!
#
# result of the measurement
#
#  x_centre  = 865.704971
#  y_centre  = 1799.600085
#  amplitude = 74.782876
#  x_fwhm    = 3.036487
#  y_fwhm    = 4.063848
#  theta     = -2.378401
#
# X_CENTRE, Y_CENTRE, FWHM
865.704971 1799.600085 3.550167
# now, generating a plot...
# finished generating a plot!
```

The star ID 16 is located at $(x, y) = (865.70, 1799.60)$ and its FWHM is 3.55 pixel. Check the image. (8)

1.8 Aperture photometry of two stars

Make a Python script to carry out aperture photometry.

Python Code 5: advobs202202_s12_05.py

```
#!/usr/pkg/bin/python3.9

#
# Time-stamp: <2022/04/21 17:08:29 (CST) daisuke>
#

# importing argparse module
import argparse

# importing sys module
import sys

# importing pathlib module
import pathlib

# importing numpy module
import numpy

# importing astropy module
import astropy.io.fits
```

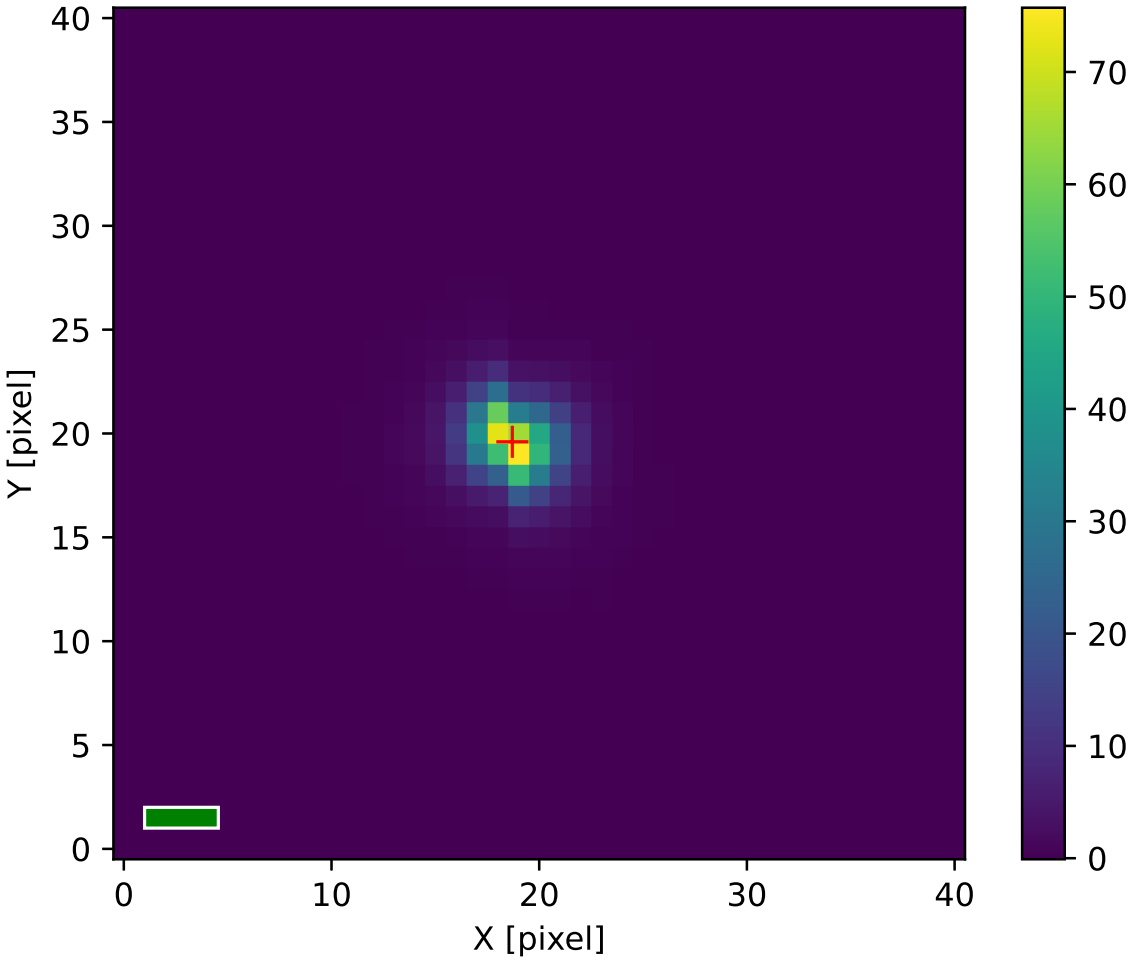

Figure 8: The results of centroid and FWHM measurement for the star ID 16.

```
import astropy.modeling
import astropy.stats

# importing photutils module
import photutils.centroids
import photutils.aperture

# importing matplotlib module
import matplotlib.figure
import matplotlib.backends.backend_agg
import matplotlib.patches

# constructing parser object
desc   = 'calculating net flux of star'
parser = argparse.ArgumentParser (description=desc)

# colour maps
choices_cmap = ['viridis', 'plasma', 'inferno', 'magma', 'cividis', \
                'binary', 'gray', 'bone', 'pink', \
                'spring', 'summer', 'autumn', 'winter', \
                'cool', 'hot', 'copper', 'ocean', 'terrain', \
                'gnuplot', 'cubehelix', 'jet', 'turbo']

# adding command-line arguments
parser.add_argument ('-f', '--fwhm', type=float, default=4.0, \
                     help='FWHM of stellar PSF in pixel (default: 4.0)')
parser.add_argument ('-a', '--aperture', type=float, default=1.5, \
                     help='aperture radius in FWHM (default: 1.5)')
parser.add_argument ('-s1', '--skyannulus1', type=float, default=3.0, \
                     help='inner sky annulus radius in FWHM (default: 3.0)')
parser.add_argument ('-s2', '--skyannulus2', type=float, default=5.0, \
                     help='outer sky annulus radius in FWHM (default: 5.0)')
parser.add_argument ('-w', '--width', type=int, default=15, \
                     help='half-width of subframe to be plotted (default: 15)')
parser.add_argument ('-x', '--xcentre', type=float, default=-1, \
                     help='x coordinate of target')
parser.add_argument ('-y', '--ycentre', type=float, default=-1, \
                     help='y coordinate of target')
parser.add_argument ('-r', '--resolution', type=int, default=450, \
                     help='resolution in DPI (default: 450)')
parser.add_argument ('-m', '--cmap', default='bone', choices=choices_cmap, \
                     help='choice of colour map (default: bone)')
parser.add_argument ('-o', '--output', default='', \
                     help='output file name')
parser.add_argument ('file', default='', help='input file name')

# command-line argument analysis
args = parser.parse_args ()

# input parameters
fwhm_pixel            = args.fwhm
aperture_radius_fwhm  = args.aperture
skyannulus_inner_fwhm = args.skyannulus1
skyannulus_outer_fwhm = args.skyannulus2
half_width            = args.width
x_centre              = args.xcentre
y_centre              = args.ycentre
resolution            = args.resolution
cmap                  = args.cmap
```

```
file_output            = args.output
file_fits              = args.file

# aperture radius and sky annulus in pixel
aperture_radius_pixel  = aperture_radius_fwhm * fwhm_pixel
skyannulus_inner_pixel = skyannulus_inner_fwhm * fwhm_pixel
skyannulus_outer_pixel = skyannulus_outer_fwhm * fwhm_pixel

# making pathlib objects
path_fits   = pathlib.Path (file_fits)

# checking input FITS file name
if (file_fits == ''):
    # printing message
    print ("You need to specify input file name.")
    # exit
    sys.exit ()
# if input file is not a FITS file, then stop the script
if not (path_fits.suffix == '.fits'):
    # printing message
    print ("Input file must be a FITS file.")
    # exit
    sys.exit ()
# if input file does not exist, then stop the script
if not (path_fits.exists ()):
    # printing message
    print ("Input file does not exist.")
    # exit
    sys.exit ()

# printing status
print ("# now, reading FITS file '%s'..." % file_fits)

# opening FITS file
with astropy.io.fits.open (file_fits) as hdu_list:
    # reading FITS header
    header = hdu_list[0].header

    # image size
    image_size_x = header['NAXIS1']
    image_size_y = header['NAXIS2']

    # checking x_centre and y_centre
    if not ( (x_centre >=0) and (x_centre < image_size_x) ):
        # printing message
        print ("Input x_centre value exceed image size.")
        # exit
        sys.exit ()
    if not ( (y_centre >=0) and (y_centre < image_size_y) ):
        print ("Input y_centre value exceed image size.")
        # printing message
        sys.exit ()
        # exit

    # reading FITS image data
    data = hdu_list[0].data

# printing status
print ("# finished reading FITS file '%s'!" % file_fits)
```

```
# printing status
print ("# now, generating an aperture...")

# making subframe
x_min = int (x_centre) - half_width
x_max = int (x_centre) + half_width + 1
y_min = int (y_centre) - half_width
y_max = int (y_centre) + half_width + 1
subframe = data[y_min:y_max, x_min:x_max]

# calculating statistical values
subframe_median = numpy.median (subframe)
subframe_stddev = numpy.std (subframe)

# position of the centre on subframe
x_centre_sub = x_centre - x_min
y_centre_sub = y_centre - y_min
position = (x_centre_sub, y_centre_sub)

# making an aperture
apphot_aperture = photutils.aperture.CircularAperture (position, \
                                                       r=aperture_radius_pixel)

# making a sky annulus
apphot_annulus \
    = photutils.aperture.CircularAnnulus (position, \
                                          r_in=skyannulus_inner_pixel, \
                                          r_out=skyannulus_outer_pixel)

# printing aperture
print ("aperture for star:")
print (apphot_aperture)

# printing sky annulus
print ("sky annulus:")
print (apphot_annulus)

# printing status
print ("# finished generating an aperture!")

# printing status
print ("# now, adding all the signal values within aperture...")

# adding all the signal values within the aperture
apphot_star \
    = photutils.aperture.aperture_photometry (subframe, apphot_aperture)

# raw_flux = star + sky
raw_flux = apphot_star['aperture_sum']

# printing result
print (apphot_star)
print ("aperture sum         = %f ADU" % raw_flux)

# printing status
print ("# finished adding all the signal values within aperture!")

# printing status
```

```
print ("# now, estimating sky background value...")

# sky background estimate
sigma_clip_4 = astropy.stats.SigmaClip (sigma=4.0, maxiters=10)
apphot_sky_stats \
    = photutils.aperture.ApertureStats (subframe, apphot_annulus, \
                                        sigma_clip=sigma_clip_4)
skybg_per_pixel     = apphot_sky_stats.mean
skybg_err_per_pixel = apphot_sky_stats.std

# printing result
print ("sky background       = %f ADU/pix" % skybg_per_pixel)
print ("sky background error = %f ADU/pix" % skybg_err_per_pixel)

# printing status
print ("# finished estimating sky background value!")

# printing status
print ("# now, subtracting sky background...")

# sky background subtraction

# net flux = (total flux within aperture)
#             - (skybg per pixel) * (number of pixels in aperture)
npix         = apphot_aperture.area
net_flux     = raw_flux - skybg_per_pixel * npix
net_flux_err = numpy.sqrt (raw_flux + npix * skybg_err_per_pixel**2)

# printing status
print ("# finished subtracting sky background!")

# printing result of aperture photometry
print ("#")
print ("# result of aperture photometry")
print ("#")
print ("# X, Y, APERTURE_RADIUS, OUTER_SKY_ANNULUS, INNER_SKY_ANNULUS,")
print ("# NPIX_APERTURE, FLUX, SKY_PER_PIX, SKY_ERR_PER_PIX,")
print ("# NET_FLUX, NET_FLUX_ERR")
print ("#")
print ("%f %f %f %f %f %f %f %f %f %f %f" \
       % (x_centre, y_centre, aperture_radius_pixel, \
          skyannulus_inner_pixel, skyannulus_outer_pixel, \
          npix, apphot_star['aperture_sum'], \
          skybg_per_pixel, skybg_err_per_pixel, \
          net_flux, net_flux_err) )

# making objects "fig" and "ax"
fig = matplotlib.figure.Figure ()
matplotlib.backends.backend_agg.FigureCanvasAgg (fig)
ax = fig.add_subplot (111)

# axes
ax.set_xlabel ('X [pixel]')
ax.set_ylabel ('Y [pixel]')

# plotting image
im = ax.imshow (subframe, origin='lower', cmap=cmap, \
                vmin=subframe_median - 3.0 * subframe_stddev, \
                vmax=subframe_median + 6.0 * subframe_stddev)
```

```
fig.colorbar (im)
ax.plot (x_centre_sub, y_centre_sub, marker='+', color='blue', markersize=10)

# adding a bar to represent FWHM
bar = matplotlib.patches.Rectangle (xy=(1,1), width=fwhm_pixel, height=1.0, \
                                    facecolor='green', edgecolor='white')
ax.add_patch (bar)

# adding a circle to represent aperture for photometry
ap = matplotlib.patches.Circle (xy=position, radius=aperture_radius_pixel, \
                                fill=False, color="yellow", linewidth=3)
ax.add_patch (ap)

# adding circles to represent sky annulus for photometry
annulus_i = matplotlib.patches.Circle (xy=position, \
                                       radius=skyannulus_inner_pixel, \
                                       fill=False, color="cyan", \
                                       linewidth=3, linestyle='--')
annulus_o = matplotlib.patches.Circle (xy=position, \
                                       radius=skyannulus_outer_pixel, \
                                       fill=False, color="cyan", \
                                       linewidth=3, linestyle='--')
ax.add_patch (annulus_i)
ax.add_patch (annulus_o)

# saving file
fig.savefig (file_output, dpi=resolution)
```

Execute the script to carry out aperture photometry of the star ID 12.

```
% ./advobs202202_s12_05.py -f 3.59 -a 1.5 -s1 3.0 -s2 5.0 -w 20 \
? -x 766.73 -y 1067.01 -o star01_phot.png pg0918_sdss_r.fits
# now, reading FITS file 'pg0918_sdss_r.fits'...
# finished reading FITS file 'pg0918_sdss_r.fits'!
# now, generating an aperture...
aperture for star:
Aperture: CircularAperture
positions: [20.73, 20.01]
r: 5.385
sky annulus:
Aperture: CircularAnnulus
positions: [20.73, 20.01]
r_in: 10.77
r_out: 17.95
# finished generating an aperture!
# now, adding all the signal values within aperture...
 id      xcenter            ycenter         aperture_sum
           pix                pix
--- ------------------ ----------------- ------------------
  1 20.730000000000018 20.00999999999999 1716.0871333986086
aperture sum         = 1716.087133 ADU
# finished adding all the signal values within aperture!
# now, estimating sky background value...
sky background       = 0.039757 ADU/pix
sky background error = 0.035116 ADU/pix
# finished estimating sky background value!
# now, subtracting sky background...
# finished subtracting sky background!
```

```
#
# result of aperture photometry
#
# X, Y, APERTURE_RADIUS, OUTER_SKY_ANNULUS, INNER_SKY_ANNULUS,
# NPIX_APERTURE, FLUX, SKY_PER_PIX, SKY_ERR_PER_PIX,
# NET_FLUX, NET_FLUX_ERR
#
766.730000 1067.010000 5.385000 10.770000 17.950000 91.100611 1716.087133 0.0397
57 0.035116 1712.465259 41.427038
```

The net flux of the star ID 12 is 1712.5 ± 41.4 ADU.
Check the sizes of aperture and sky annulus. (9)

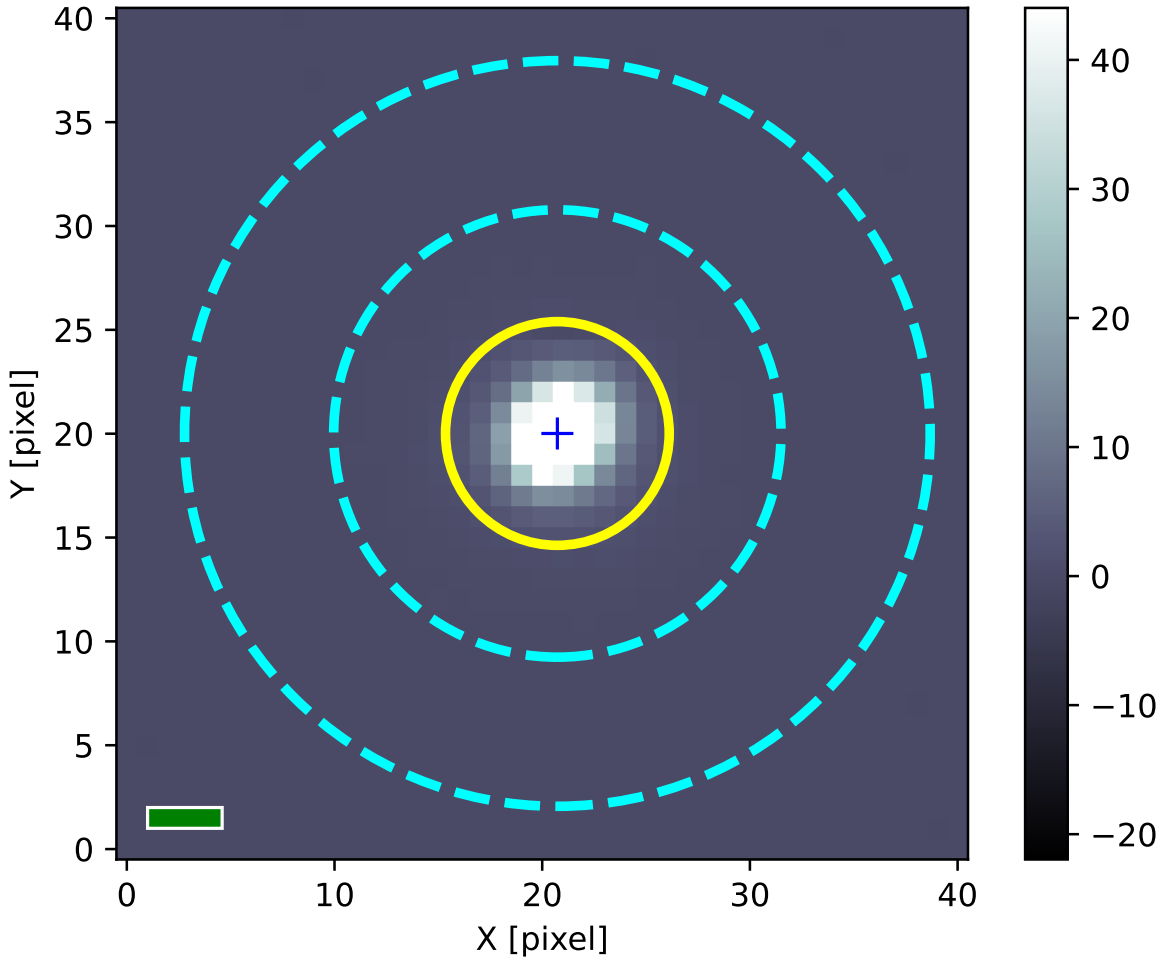

Figure 9: The aperture and sky annulus for the aperture photometry of the star ID 12.

Next, carry out aperture photometry of the star ID 16.

```
% ./advobs202202_s12_05.py -f 3.55 -a 1.5 -s1 3.0 -s2 5.0 -w 20 \
? -x 865.70 -y 1799.60 -o star02_phot.png pg0918_sdss_r.fits
# now, reading FITS file 'pg0918_sdss_r.fits'...
# finished reading FITS file 'pg0918_sdss_r.fits'!
# now, generating an aperture...
```

```
aperture for star:
Aperture: CircularAperture
positions: [20.7, 20.6]
r: 5.324999999999999
sky annulus:
Aperture: CircularAnnulus
positions: [20.7, 20.6]
r_in: 10.649999999999999
r_out: 17.75
# finished generating an aperture!
# now, adding all the signal values within aperture...
 id      xcenter            ycenter         aperture_sum
           pix                pix
--- ------------------ ----------------- ------------------
  1 20.700000000000045 20.59999999999991 1108.2513151446276
aperture sum         = 1108.251315 ADU
# finished adding all the signal values within aperture!
# now, estimating sky background value...
sky background       = 0.027684 ADU/pix
sky background error = 0.031407 ADU/pix
# finished estimating sky background value!
# now, subtracting sky background...
# finished subtracting sky background!
#
# result of aperture photometry
#
# X, Y, APERTURE_RADIUS, OUTER_SKY_ANNULUS, INNER_SKY_ANNULUS,
# NPIX_APERTURE, FLUX, SKY_PER_PIX, SKY_ERR_PER_PIX,
# NET_FLUX, NET_FLUX_ERR
#
865.700000 1799.600000 5.325000 10.650000 17.750000 89.081823 1108.251315 0.0276
84 0.031407 1105.785210 33.291728
```

The net flux of the star ID 16 is 1105.8 ± 33.3 ADU.

Check the sizes of aperture and sky annulus. (10)

1.9 Calculating magnitude of the star ID 16

Suppose the magnitude of the star ID 12 is known, and calculate the magnitude of the star ID 16 using net flux and magnitude of the star ID 12 and net flux of the star ID 16.

The r'-band magnitude of the star ID 12 is 14.339.

```
012  09h21m35.1168s    +02d46m19.56s     14.339
```

Make a Python script to calculate r'-band magnitude of the star ID 16.

Python Code 6: advobs202202_s12_06.py

```
#!/usr/pkg/bin/python3.9

#
# Time-stamp: <2022/04/21 17:23:06 (CST) daisuke>
#

# importing math module
import math

# r'-band magnitude of star ID 12
```

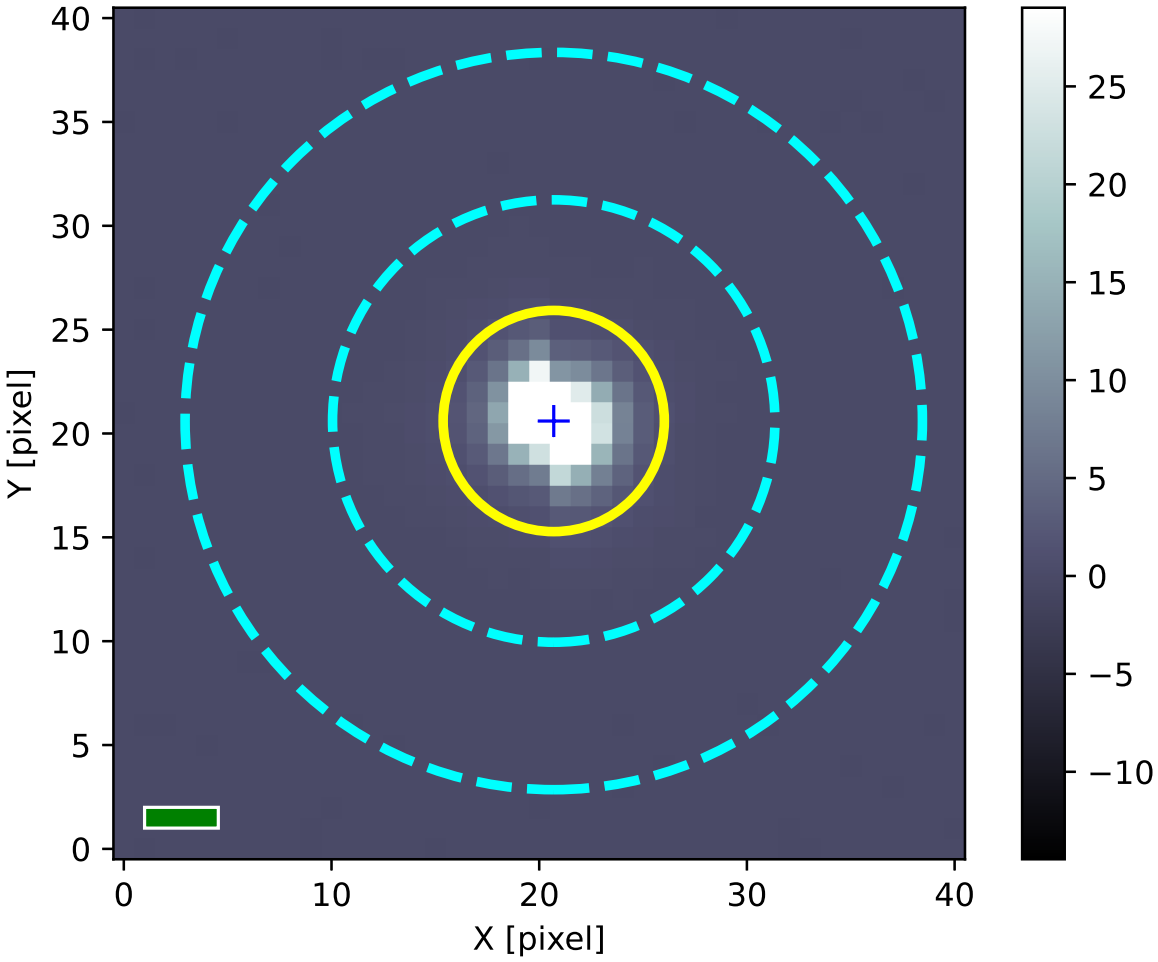

Figure 10: The aperture and sky annulus for the aperture photometry of the star ID 16.

```
mag_star1 = 14.339

# net flux of star ID 12
flux_star1 = 1712.5

# flux error of star ID 12
err_star1 = 41.4

# net flux of star ID 16
flux_star2 = 1105.8

# flux error of star ID 16
err_star2 = 33.3

# r'-band magnitude of star ID 16
mag_star2 = mag_star1 - 2.5 * math.log10 (flux_star2 / flux_star1)

# error on magnitude
magerr_star1 = 2.5 * math.log10 (1 + err_star1 / flux_star1)
magerr_star2 = 2.5 * math.log10 (1 + err_star2 / flux_star2)
magerr_total = math.sqrt (magerr_star1**2 + magerr_star2**2)

# printing result
print ("#")
print ("# input parameters")
print ("#")
print ("#  mag_star1  = %f" % mag_star1)
print ("#  flux_star1 = %f ADU" % flux_star1)
print ("#  err_star1  = %f ADU" % err_star1)
print ("#  flux_star2 = %f ADU" % flux_star2)
print ("#  err_star2  = %f ADU" % err_star2)
print ("#")
print ("mag_star2 = %f +/- %f" % (mag_star2, magerr_total) )
```

Execute the script.

```
% chmod a+x advobs202202_s12_06.py
% ./advobs202202_s12_06.py
#
# input parameters
#
#  mag_star1  = 14.339000
#  flux_star1 = 1712.500000 ADU
#  err_star1  = 41.400000 ADU
#  flux_star2 = 1105.800000 ADU
#  err_star2  = 33.300000 ADU
#
mag_star2 = 14.813885 +/- 0.041356
```

The result of the calculation is 14.81 ± 0.04. It is consistent with the catalogue value of 14.839.

1.10 Error propagation using `uncertainties` package

The error propagation is calculated by using `uncertainties` package. Visit the official website of `uncertainties` package and learn about it. If you do not have `uncertainties` package on your computer, install it.

- `https://pythonhosted.org/uncertainties/` (Fig. 11)
- `https://uncertainties-python-package.readthedocs.io/`

- `https://pypi.org/project/uncertainties/`
- `https://github.com/lebigot/uncertainties`

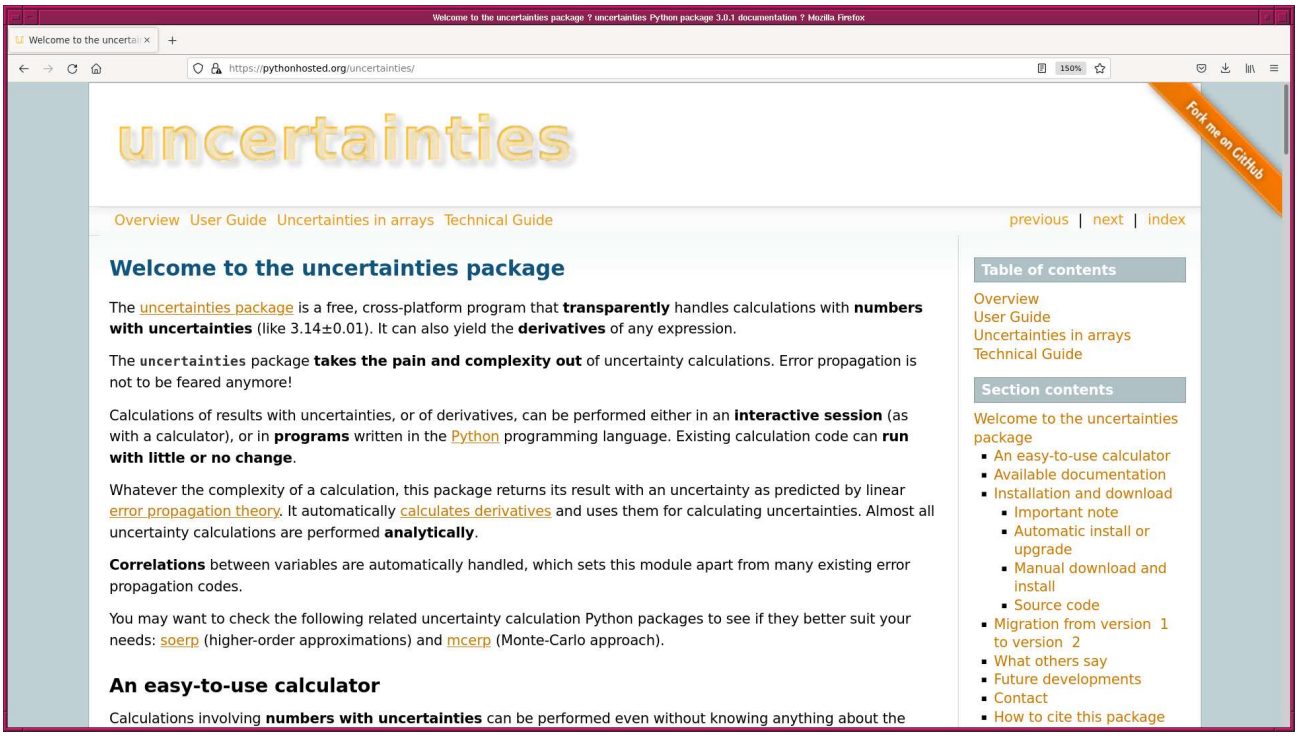

Figure 11: The official website of `uncertainties` package.

Make a Python script to calculate magnitude of the star ID 16 using `uncertainties` package.

Python Code 7: advobs202202_s12_07.py

```
#!/usr/pkg/bin/python3.9

#
# Time-stamp: <2022/04/21 20:48:17 (CST) daisuke>
#

# importing math module
import math

# importing uncertainties module
import uncertainties
import uncertainties.umath

# r'-band magnitude of star ID 12
mag_star1 = 14.339

# net flux of star ID 12
# 1712.5 +/- 41.4
flux_star1 = uncertainties.ufloat (1712.5, 41.4)

# net flux of star ID 16
# 1105.8 +/- 33.3
flux_star2 = uncertainties.ufloat (1105.8, 33.3)
```

```
# r'-band magnitude of star ID 16
mag_star2 \
    = mag_star1 - 2.5 * uncertainties.umath.log10 (flux_star2 / flux_star1)

# printing result
print ("#")
print ("# input parameters")
print ("#")
print ("#  mag_star1  =", mag_star1)
print ("#  flux_star1 =", flux_star1)
print ("#  flux_star2 =", flux_star2)
print ("#")
print ("mag_star2 =", mag_star2)
```

Execute the script.

```
% chmod a+x advobs202202_s12_07.py
% ./advobs202202_s12_07.py
#
# input parameters
#
#  mag_star1  = 14.339
#  flux_star1 = (1.71+/-0.04)e+03
#  flux_star2 = 1106+/-33
#
mag_star2 = 14.81+/-0.04
```

2 Photometry of stars on image obtained at Lulin

Next, we carry out photometry of stars on image obtained at Lulin Observatory.

2.1 Photometric standard stars in PG1047+003 field

Show a list of photometric standard stars in PG1047+003 field.

```
% ./advobs202202_s12_01.py stds/PG1047+003A.txt.clean.v3
# star ID, RA, Dec, r'-band mag.
004  10h50m29.6904s    +00d04m21s        11.709
005  10h50m13.6992s    -00d00m32.4s      12.373
006  10h50m05.6712s    -00d01m11.28s     13.304
007  10h50m02.8416s    -00d00m36.72s     13.718
008  10h50m23.0376s    -00d04m54.48s     13.803
010  10h49m53.2776s    -00d03m34.56s     14.308
011  10h49m52.9968s    -00d03m28.44s     14.318
013  10h50m07.932s     -00d02m04.56s     14.553
014  10h50m07.884s     -00d05m35.52s     14.616
017  10h50m17.8608s    -00d01m14.88s     14.838
020  10h50m28.5888s    +00d01m46.2s      14.897
022  10h50m14.232s     -00d05m27.24s     14.933
025  10h49m59.8224s    -00d02m35.88s     15.322
```

2.2 Downloading DSS image of PG1047+003 field

Download DSS image of PG1047+003 field.

```
% ./advobs202202_s12_02.py -r simbad -s "DSS2 Red" -t PG1047+003 -f 1536 \
? -o pg1047_dss_r.fits
Target Name: PG1047+003
  RA:  10h50m02.8264s = 162.511777 deg
  Dec: -00d00m36.88s = -0.010244 deg
Available images:
  ['https://skyview.gsfc.nasa.gov/tempspace/fits/skv20222567820259.fits']
Downloading https://skyview.gsfc.nasa.gov/tempspace/fits/skv4810443363550.fits
|=========================================| 9.4M/9.4M (100.00%)        21s
% ls -lF *.fits
-rw-r--r--  1 daisuke  taiwan  16793280 Apr 21 14:32 pg0918_sdss_r.fits
-rw-r--r--  1 daisuke  taiwan   9449280 Apr 21 21:29 pg1047_dss_r.fits
```

2.3 Visualisation of DSS image

Use Ginga to visualise the DSS image. (Fig. 12)

```
% ginga pg1047_dss_r.fits
```

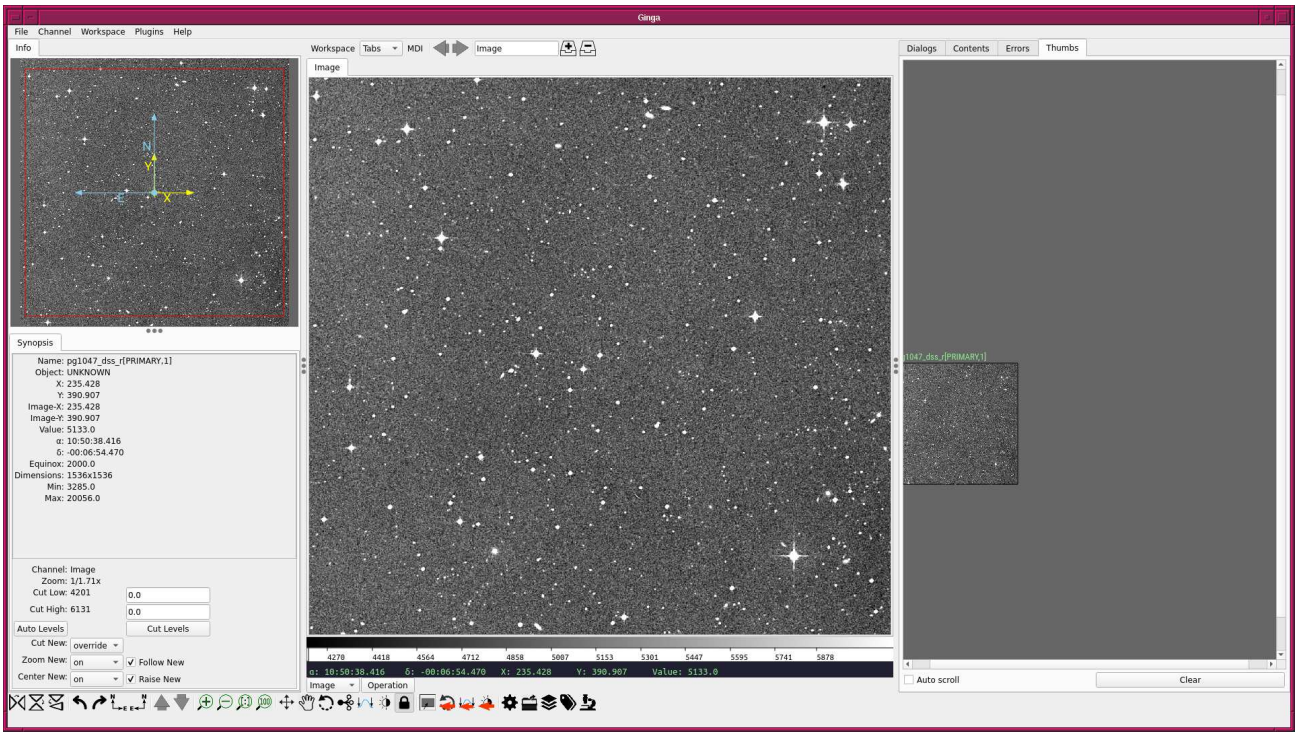

Figure 12: DSS image around the star PG1047+003.

2.4 Show the positions of standard stars

Show the positions of standard stars in PG1047+003 field.

```
% ./advobs202202_s12_03.py -s stds/PG1047+003A.txt.clean.v3 -c viridis \
? -o pg1047_std.png pg1047_dss_r.fits
% ls -lF pg1047_std.png
-rw-r--r--  1 daisuke  taiwan  8118663 Apr 21 21:35 pg1047_std.png
```

Show the image file created by the script. (Fig. 13)

```
% feh -dF pg1047_std.png
```

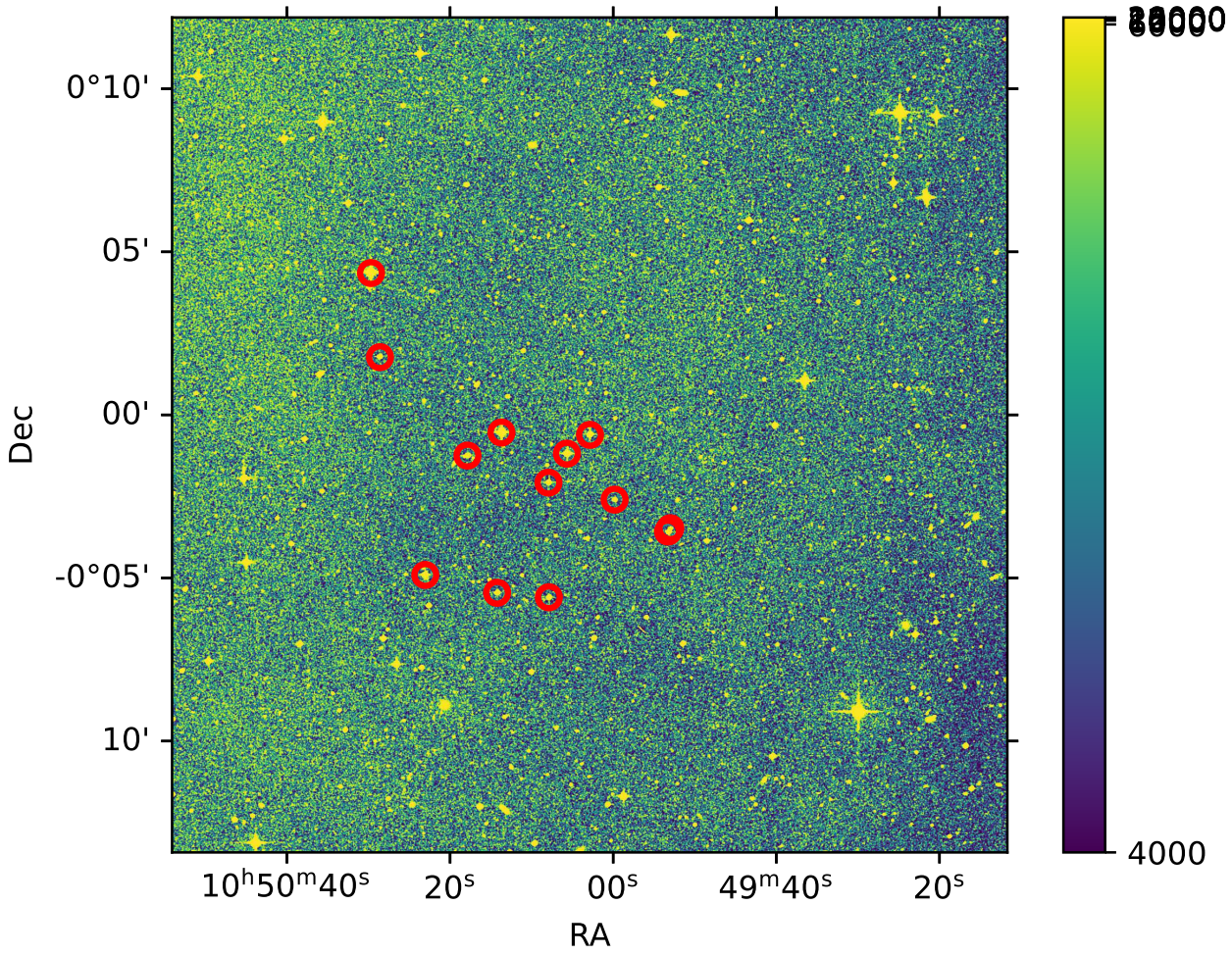

Figure 13: Locations of photometric standard stars in PG1047+003 field.

2.5 Choosing two stars

Pick two stars for your measurements. Choose two stars on the image `pg1047_std.png`. Then, use Ginga to find rough values of RA and Dec of those two stars. Then, find those two stars on the standard star file.

For example, following two stars are selected. (Fig. 14 and 15) These are stars of ID 7 and 13.

```
007  10h50m02.8416s    -00d00m36.72s     13.718
013  10h50m07.932s     -00d02m04.56s     14.553
```

2.6 Identifying stars in our image

We have reduced the images taken on 15 February 2021 at the session 08. For following analysis, we use the image `lot_20210215_0116_df.fits`. Use Ginga to visualise the image. (Fig. 16) Push the button to make the orientation of the image to be "north is up and east is left".

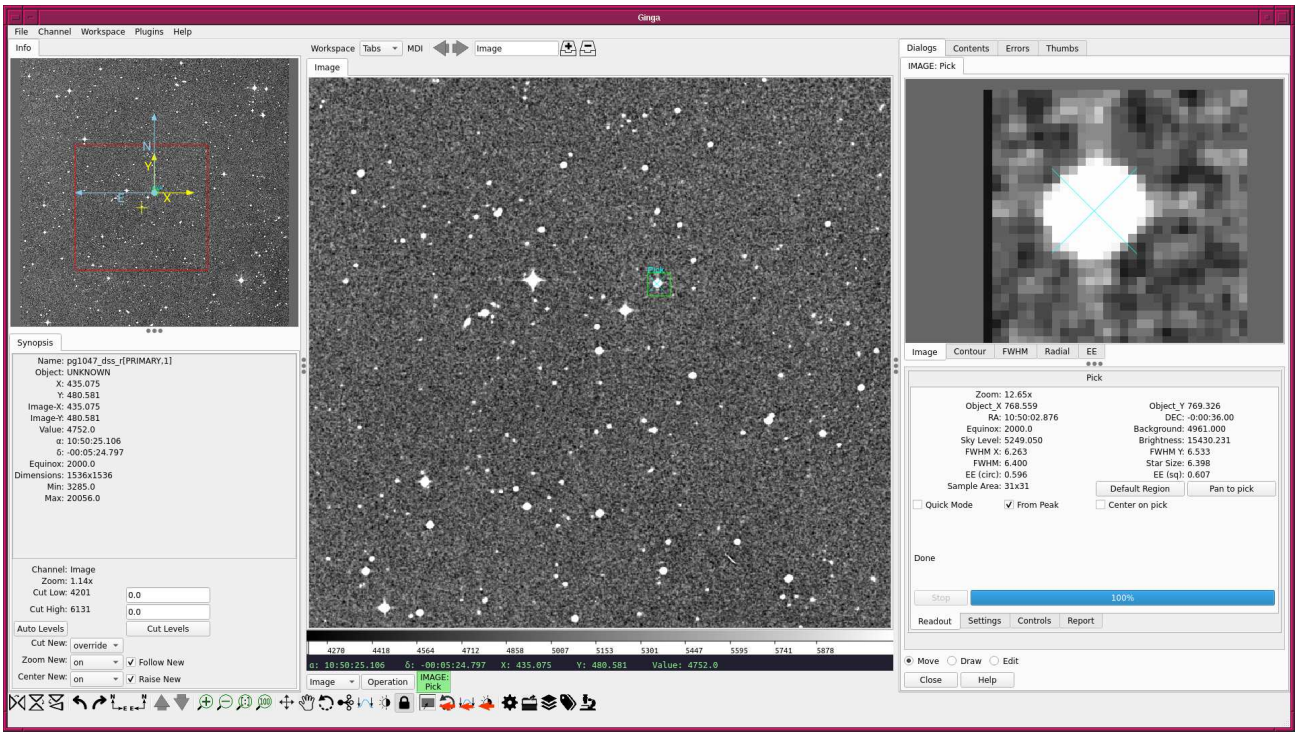

Figure 14: First star for the measurement.

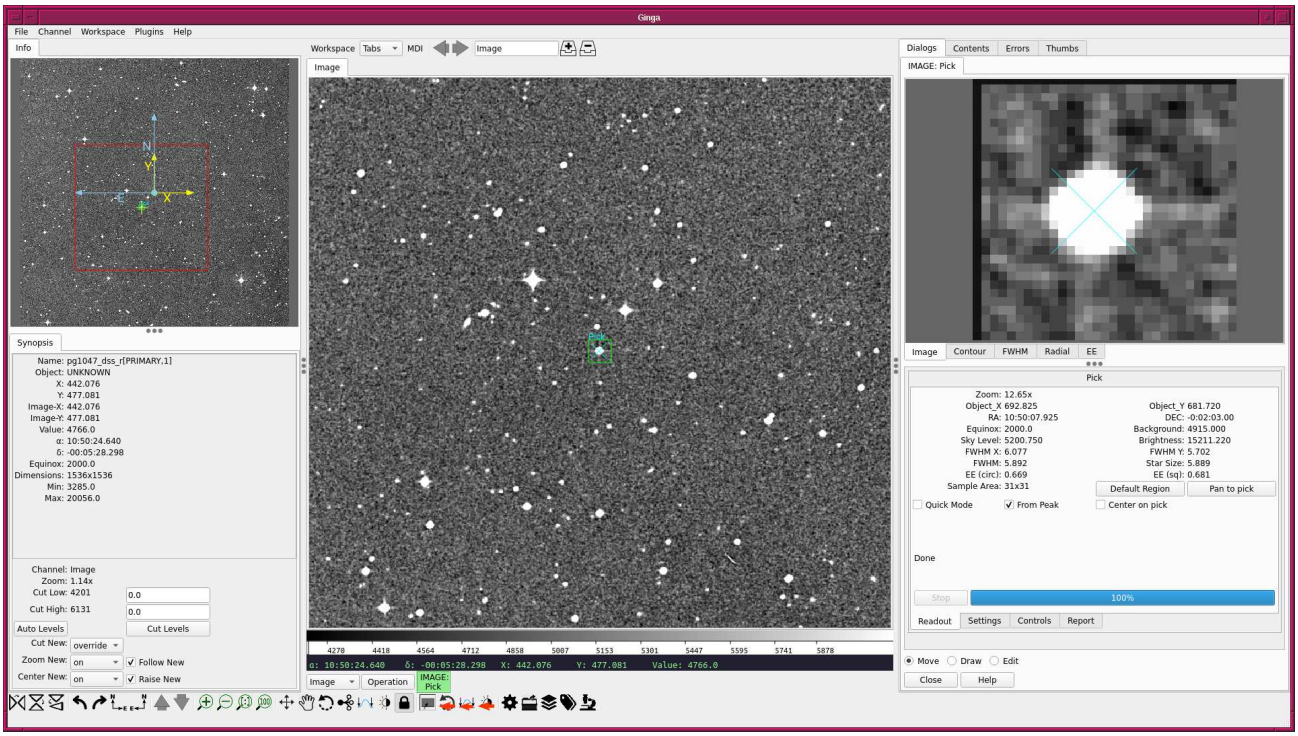

Figure 15: Second star for the measurement.

```
% CAP -pi /somewhere/in/the/disk/lot_20210215_0116_df.fits .
% ls -lF lot_20210215_0116_df.fits
-rw-r--r--  1 daisuke  taiwan  33566400 Apr  8 10:39 lot_20210215_0116_df.fits
% ginga lot_20210215_0116_df.fits
```

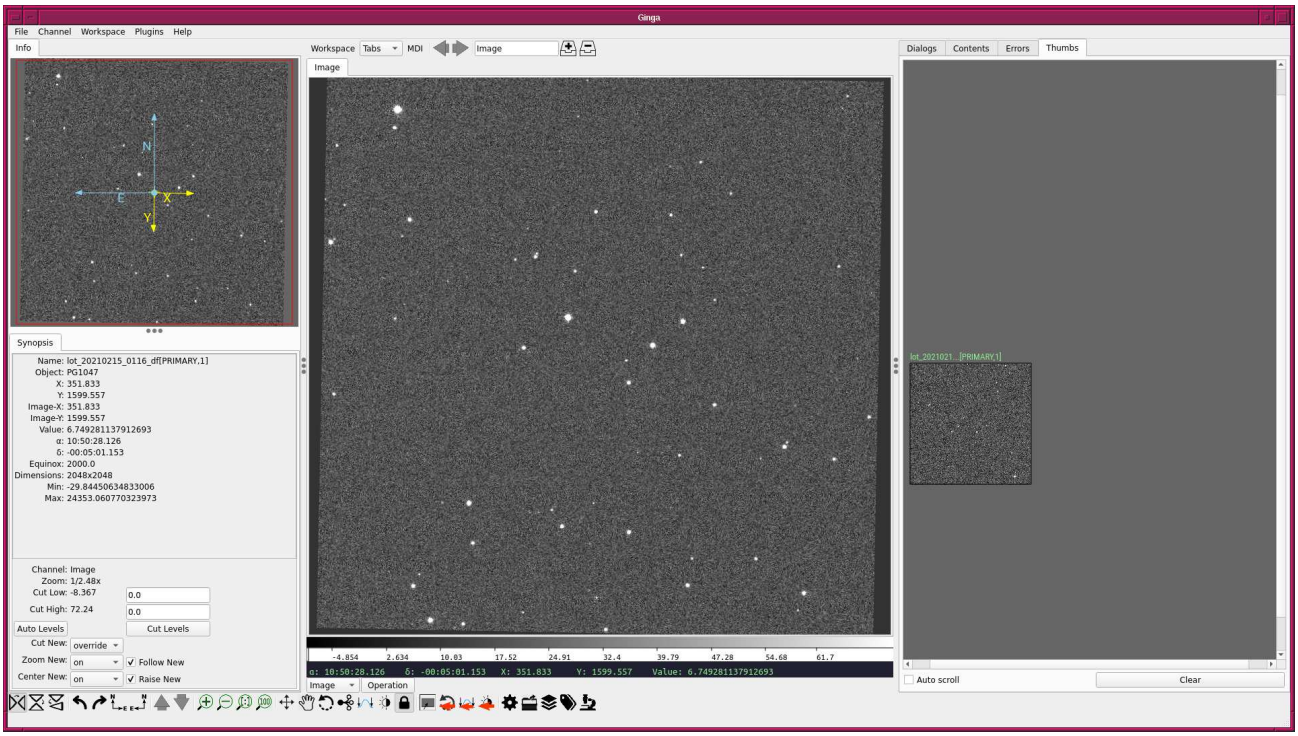

Figure 16: The image of `lot_20210215_0116_df.fits`.

Measure rough (x, y) positions of two stars on our image. The star of ID 7 is at (x, y) $\sim$ (1325, 896) and ID 13 is at (x, y) $\sim$ (1130, 1125). (Fig. 17 and 18)

2.7 Measuring accurate position of stars using centroid

Measure centroid and FWHM of two stars.

```
% ./advobs202202_s12_04.py -c 2dg -p 2dg -w 35 -x 1325 -y 896 \
? -o lot_star01_centroid.png lot_20210215_0116_df.fits
#
# input parameters
#
#  input file name           = lot_20210215_0116_df.fits
#  half-width of search box  = 35.000000
#  x_init                    = 1325.000000
#  y_init                    = 896.000000
#  centroid technique        = 2dg
#  output file name          = lot_star01_centroid.png
#
# now, reading FITS file...
# finished reading FITS file!
# now, extracting image around the target object...
# finished extracting image around the target object!
# now, subtracting background...
# finished subtracting background!
```

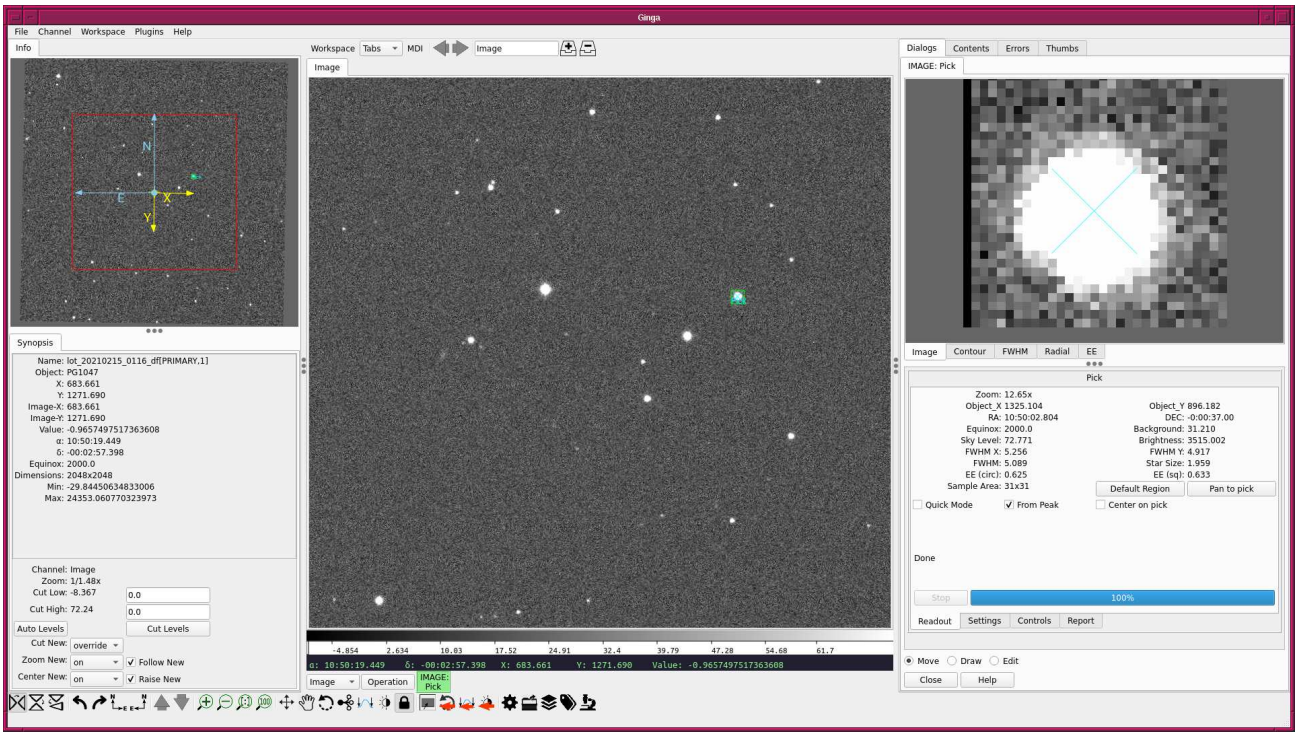

Figure 17: The location of star ID 7 on the image of `lot_20210215_0116_df.fits`.

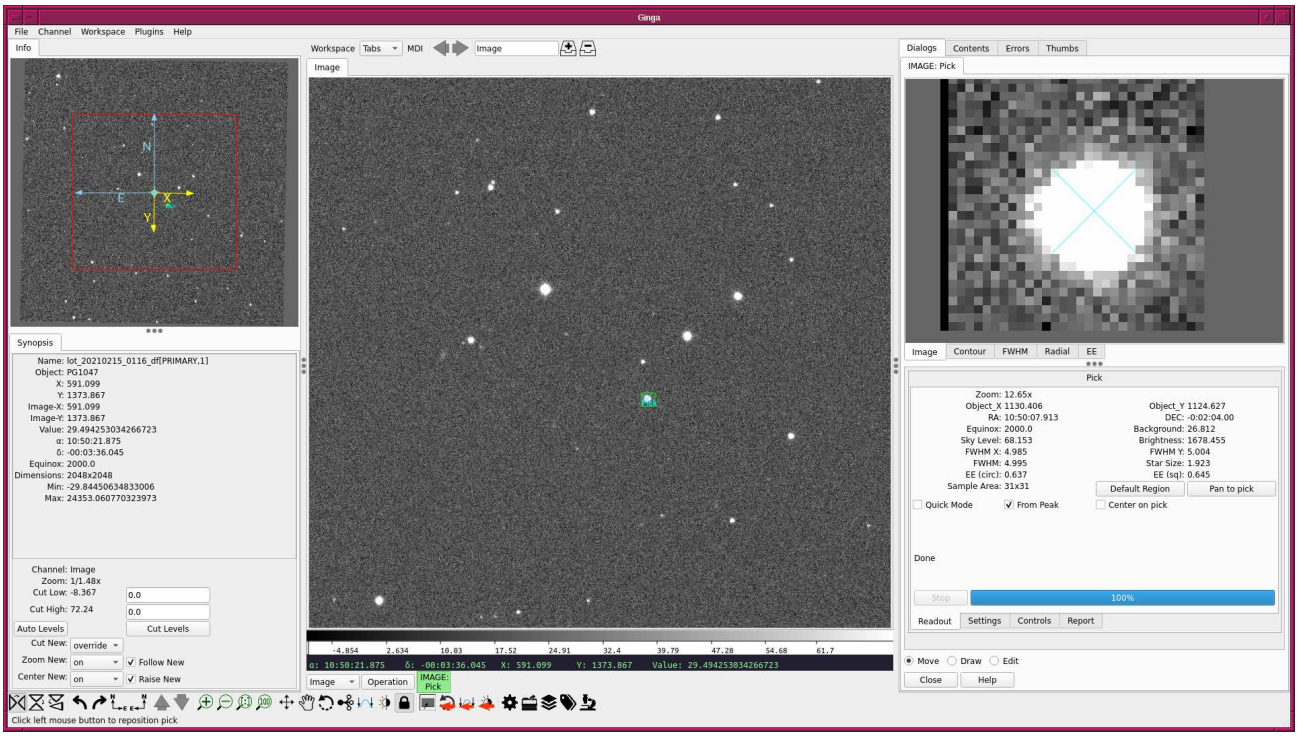

Figure 18: The location of star ID 13 on the image of `lot_20210215_0116_df.fits`.

```
# now, measuring centroid...
# finished measuring!
# now, measuring PSF...
# finished measuring PSF!
#
# result of the measurement
#
#  x_centre  = 1323.966920
#  y_centre  = 895.113399
#  amplitude = 3432.808295
#  x_fwhm    = 5.366112
#  y_fwhm    = 5.068981
#  theta     = -3.236540
#
# X_CENTRE, Y_CENTRE, FWHM
1323.966920 895.113399 5.217546
# now, generating a plot...
# finished generating a plot!
% ls -lF lot_star01_centroid.png
-rw-r--r--  1 daisuke  taiwan  129130 Apr 21 22:03 lot_star01_centroid.png
```

The star ID 7 is located at (x, y) = (1323.97, 895.11) and its FWHM is 5.22 pixel. (Fig. 19)
Next, measure centroid and FWHM of star ID 13.

```
% ./advobs202202_s12_04.py -c 2dg -p 2dg -w 35 -x 1130 -y 1125 \
? -o lot_star02_centroid.png lot_20210215_0116_df.fits
#
# input parameters
#
#  input file name          = lot_20210215_0116_df.fits
#  half-width of search box = 35.000000
#  x_init                   = 1130.000000
#  y_init                   = 1125.000000
#  centroid technique       = 2dg
#  output file name         = lot_star02_centroid.png
#
# now, reading FITS file...
# finished reading FITS file!
# now, extracting image around the target object...
# finished extracting image around the target object!
# now, subtracting background...
# finished subtracting background!
# now, measuring centroid...
# finished measuring!
# now, measuring PSF...
# finished measuring PSF!
#
# result of the measurement
#
#  x_centre  = 1129.322355
#  y_centre  = 1123.505496
#  amplitude = 1639.084914
#  x_fwhm    = 5.239267
#  y_fwhm    = 5.087630
#  theta     = 0.279134
#
# X_CENTRE, Y_CENTRE, FWHM
1129.322355 1123.505496 5.163449
```

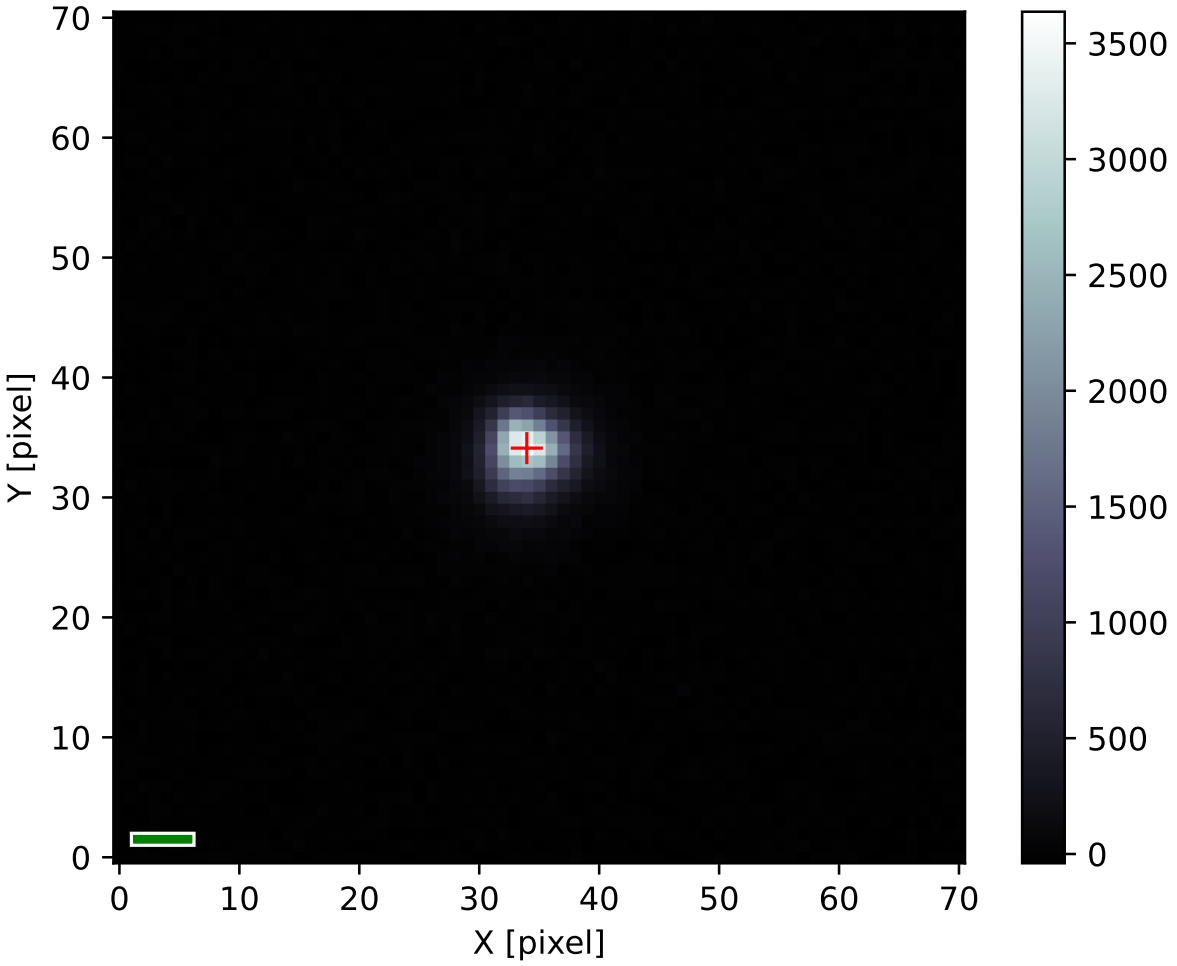

Figure 19: The centroid measurement of the star ID 7.

```
# now, generating a plot...
# finished generating a plot!
% ls -lF lot_star0*.png
-rw-r--r--  1 daisuke  taiwan  129130 Apr 21 22:03 lot_star01_centroid.png
-rw-r--r--  1 daisuke  taiwan  137759 Apr 21 22:04 lot_star02_centroid.png
```

The star ID 13 is located at (x, y) = (1129.32, 1123.51) and its FWHM is 5.16 pixel. (Fig. 20)

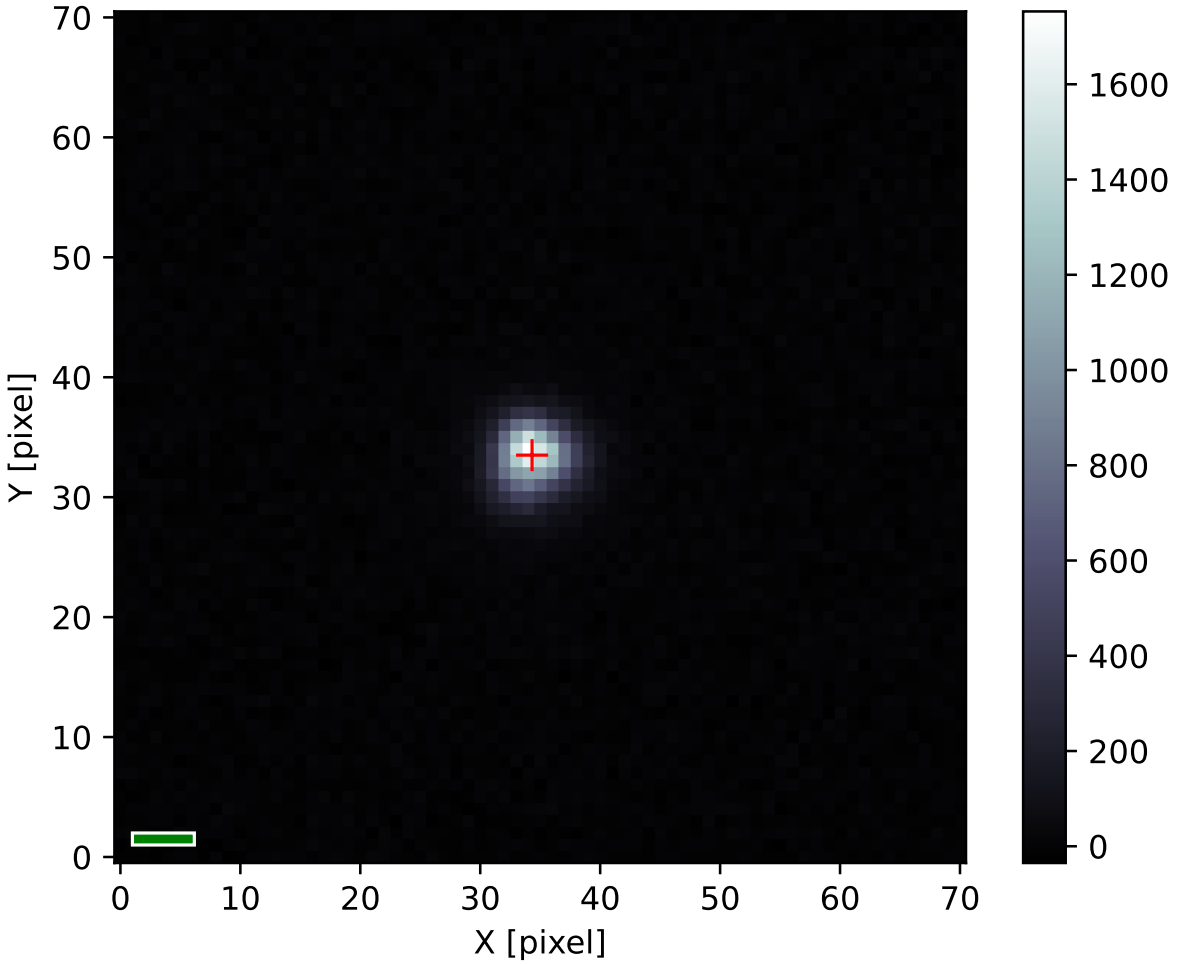

Figure 20: The centroid measurement of the star ID 13.

2.8 Aperture photometry of two stars

Measure the brightness of the star ID 7.

```
% ./advobs202202_s12_05.py -f 5.22 -a 1.5 -s1 3.0 -s2 6.0 -w 35 \
? -x 1323.97 -y 895.11 -o lot_star01_aperture.png lot_20210215_0116_df.fits
# now, reading FITS file 'lot_20210215_0116_df.fits'...
# finished reading FITS file 'lot_20210215_0116_df.fits'!
# now, generating an aperture...
aperture for star:
Aperture: CircularAperture
```

```
positions: [35.97, 35.11]
r: 7.83
sky annulus:
Aperture: CircularAnnulus
positions: [35.97, 35.11]
r_in: 15.66
r_out: 31.32
# finished generating an aperture!
# now, adding all the signal values within aperture...
 id      xcenter           ycenter          aperture_sum
           pix               pix
--- ----------------- ------------------ ------------------
  1 35.97000000000003 35.110000000000014 115995.17308938605
aperture sum         = 115995.173089 ADU
# finished adding all the signal values within aperture!
# now, estimating sky background value...
sky background       = 18.943507 ADU/pix
sky background error = 9.344889 ADU/pix
# finished estimating sky background value!
# now, subtracting sky background...
# finished subtracting sky background!
#
# result of aperture photometry
#
# X, Y, APERTURE_RADIUS, OUTER_SKY_ANNULUS, INNER_SKY_ANNULUS,
# NPIX_APERTURE, FLUX, SKY_PER_PIX, SKY_ERR_PER_PIX,
# NET_FLUX, NET_FLUX_ERR
#
1323.970000 895.110000 7.830000 15.660000 31.320000 192.607590 115995.173089 18.
943507 9.344889 112346.509887 364.437931
```

The net flux of star ID 7 is 112347 ± 364 ADU.
Next, measure the brightness of star ID 13.

```
% ./advobs202202_s12_05.py -f 5.16 -a 1.5 -s1 3.0 -s2 6.0 -w 35 \
? -x 1129.32 -y 1123.51 -o lot_star02_aperture.png lot_20210215_0116_df.fits
# now, reading FITS file 'lot_20210215_0116_df.fits'...
# finished reading FITS file 'lot_20210215_0116_df.fits'!
# now, generating an aperture...
aperture for star:
Aperture: CircularAperture
positions: [35.32, 35.51]
r: 7.74
sky annulus:
Aperture: CircularAnnulus
positions: [35.32, 35.51]
r_in: 15.48
r_out: 30.96
# finished generating an aperture!
# now, adding all the signal values within aperture...
 id      xcenter            ycenter         aperture_sum
           pix                pix
--- ------------------ ----------------- ----------------
  1 35.319999999999936 35.50999999999999 55854.4503916894
aperture sum         = 55854.450392 ADU
# finished adding all the signal values within aperture!
# now, estimating sky background value...
sky background       = 18.614628 ADU/pix
```

```
sky background error = 9.101136 ADU/pix
# finished estimating sky background value!
# now, subtracting sky background...
# finished subtracting sky background!
#
# result of aperture photometry
#
# X, Y, APERTURE_RADIUS, OUTER_SKY_ANNULUS, INNER_SKY_ANNULUS,
# NPIX_APERTURE, FLUX, SKY_PER_PIX, SKY_ERR_PER_PIX,
# NET_FLUX, NET_FLUX_ERR
#
1129.320000 1123.510000 7.740000 15.480000 30.960000 188.205276 55854.450392 18.
614628 9.101136 52351.079101 267.289396
```

The net flux of star ID 13 is 52351 ± 267 ADU.
Apertures and sky annuli for stars ID 7 and ID 13 are shown in Fig. **??** and **??**.

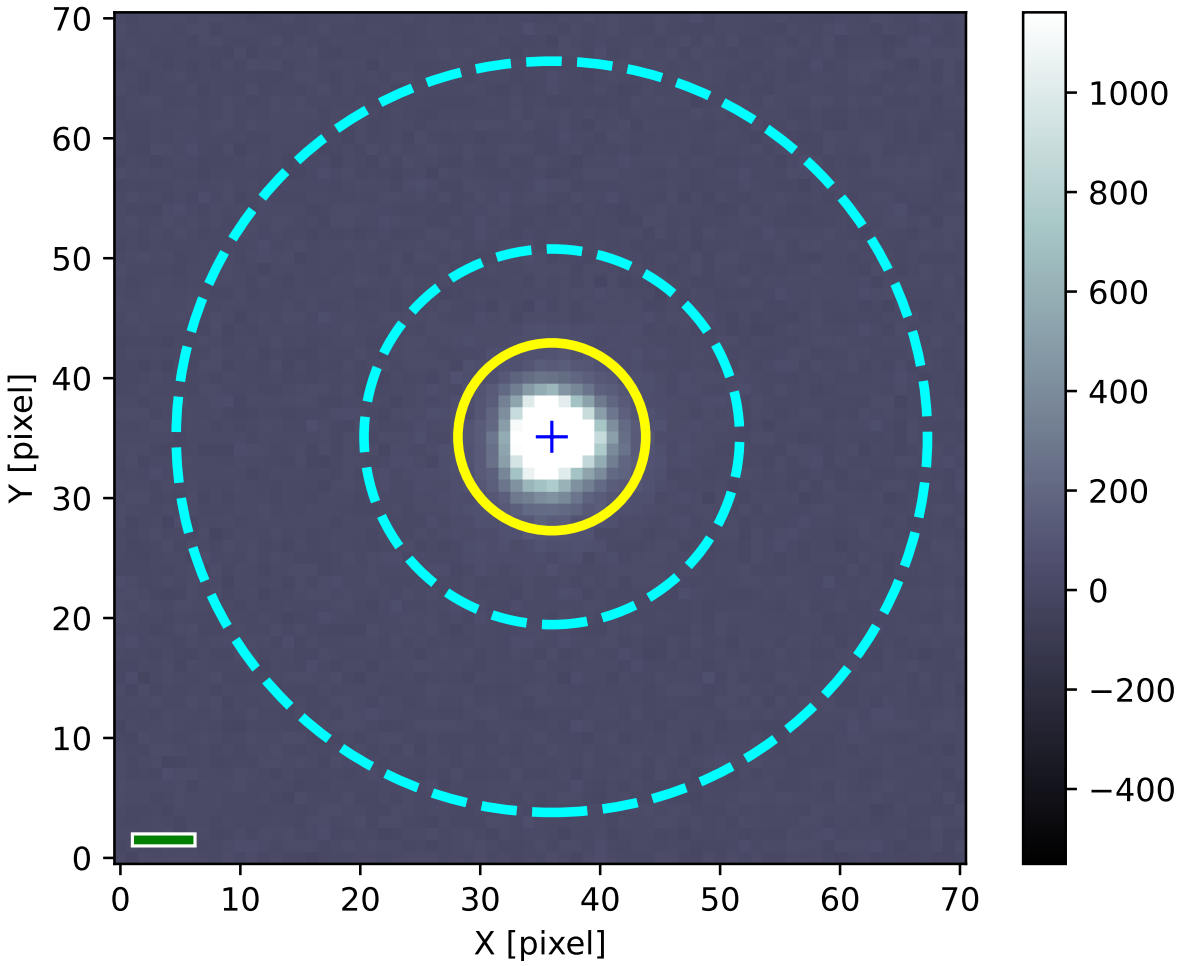

Figure 21: Aperture and sky annulus set for star ID 7.

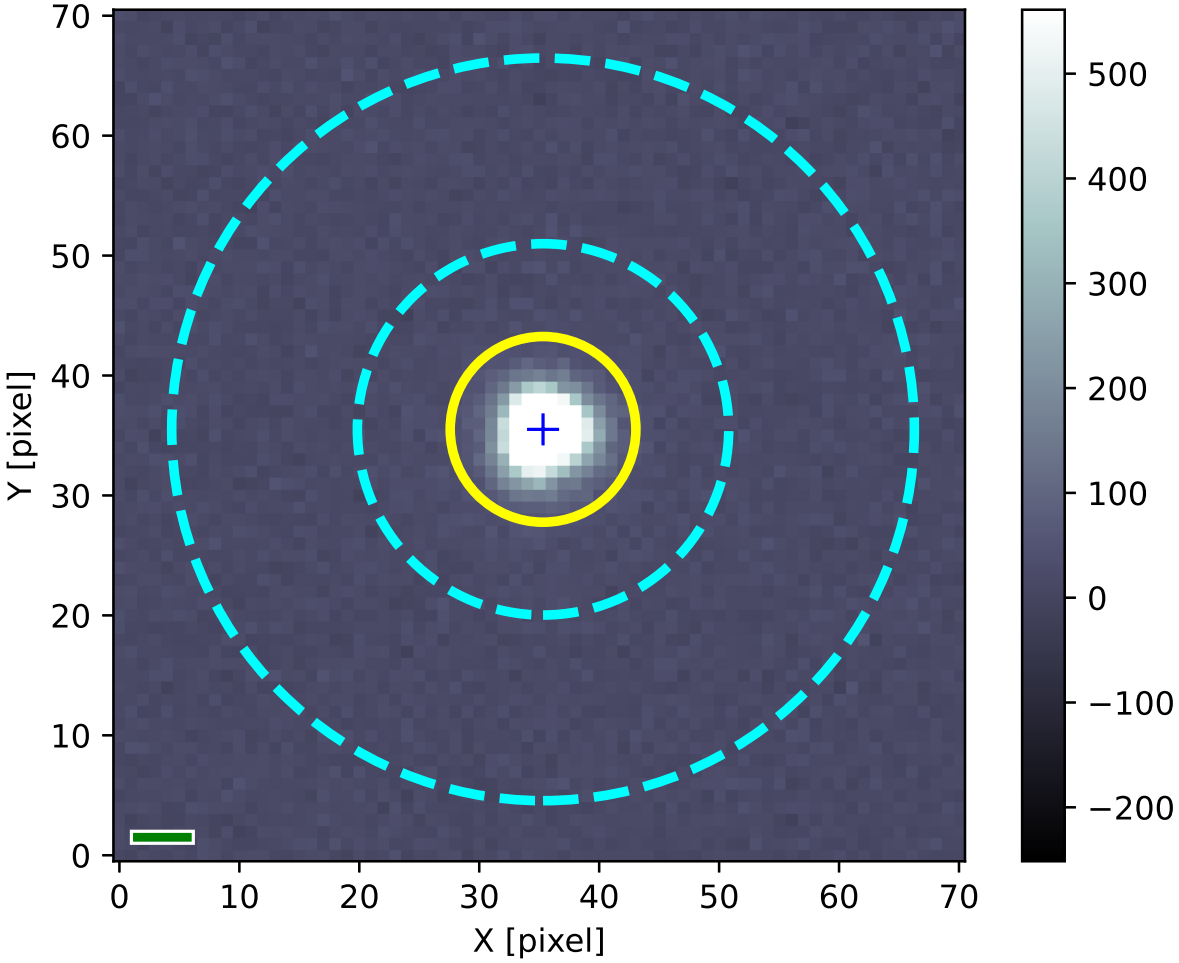

Figure 22: Aperture and sky annulus set for star ID 13.

2.9 Calculating magnitude of a star

The r'-band magnitude of star ID 7 is 13.718. From aperture photometry, net fluxes of two stars are 112347 ± 364 ADU and 52351 ± 267 ADU, respectively. Use these information to calculate the magnitude of star ID 13.

Python Code 8: advobs202202_s12_08.py

```
#!/usr/pkg/bin/python3.9

#
# Time-stamp: <2022/04/21 22:24:31 (CST) daisuke>
#

# importing math module
import math

# r'-band magnitude of star ID 7
mag_star1 = 13.718

# net flux of star ID 7
flux_star1 = 112347

# flux error of star ID 7
err_star1 = 364

# net flux of star ID 13
flux_star2 = 52351

# flux error of star ID 13
err_star2 = 267

# r'-band magnitude of star ID 13
mag_star2 = mag_star1 - 2.5 * math.log10 (flux_star2 / flux_star1)

# error on magnitude
magerr_star1 = 2.5 * math.log10 (1 + err_star1 / flux_star1)
magerr_star2 = 2.5 * math.log10 (1 + err_star2 / flux_star2)
magerr_total = math.sqrt (magerr_star1**2 + magerr_star2**2)

# printing result
print ("#")
print ("# input parameters")
print ("#")
print ("#  mag_star1  = %f" % mag_star1)
print ("#  flux_star1 = %f ADU" % flux_star1)
print ("#  err_star1  = %f ADU" % err_star1)
print ("#  flux_star2 = %f ADU" % flux_star2)
print ("#  err_star2  = %f ADU" % err_star2)
print ("#")
print ("mag_star2 = %f +/- %f" % (mag_star2, magerr_total) )
```

Run the script and calculate r'-band magnitude of the star ID 13.

```
% chmod a+x advobs202202_s12_08.py
% ./advobs202202_s12_08.py
#
# input parameters
#
#  mag_star1  = 13.718000
```

```
#  flux_star1 = 112347.000000 ADU
#  err_star1  = 364.000000 ADU
#  flux_star2 = 52351.000000 ADU
#  err_star2  = 267.000000 ADU
#
mag_star2 = 14.547091 +/- 0.006545
```

The magnitude of star ID 13 is calculated to be 14.547 ± 0.007. From the photometric standard star catalogue, r'-band magnitude of the star ID 13 is 14.553 ± 0.005. Our measurement is consistent with the catalogue value.

2.10 Calculation using `uncertainties` package

Use `uncertainties` package to calculate the error.

Python Code 9: advobs202202_s12_09.py

```
#!/usr/pkg/bin/python3.9

#
# Time-stamp: <2022/04/21 22:21:53 (CST) daisuke>
#

# importing math module
import math

# importing uncertainties module
import uncertainties
import uncertainties.umath

# r'-band magnitude of star ID 12
mag_star1 = 13.718

# net flux of star ID 7
# 112347 +/- 364
flux_star1 = uncertainties.ufloat (112347, 364)

# net flux of star ID 13
# 52351 +/- 267
flux_star2 = uncertainties.ufloat (52351, 267)

# r'-band magnitude of star ID 16
mag_star2 \
    = mag_star1 - 2.5 * uncertainties.umath.log10 (flux_star2 / flux_star1)

# printing result
print ("#")
print ("# input parameters")
print ("#")
print ("#  mag_star1  =", mag_star1)
print ("#  flux_star1 =", flux_star1)
print ("#  flux_star2 =", flux_star2)
print ("#")
print ("mag_star2 =", mag_star2)
```

Execute the script.

```
% chmod a+x advobs202202_s12_09.py
% ./advobs202202_s12_09.py
```

```
#
# input parameters
#
#  mag_star1  = 13.718
#  flux_star1 = (1.123+/-0.004)e+05
#  flux_star2 = (5.235+/-0.027)e+04
#
mag_star2 = 14.547+/-0.007
```

We have obtained the same result.

3 For your further reading

1. Read the chapter 4 "Photometric Concepts and Magnitudes" of the textbook "Fundamental Astronomy".
   - "Fundamental Astronomy"
     - Karttunen, H., Kroger, P., Oja, H., Poutanen, M., Donner, K.J.
     - Springer
     - 2017 (6th edition)
     - `https://www.springer.com/gp/book/9783662530443`
2. Read following paper to learn about uncertainty analysis.
   - "Uncertainty Analysis in Photometric Observations"
     - Koppelman, M.
     - 2005
     - `http://adsabs.harvard.edu/full/2005SASS...24..107K`

4 Exercise

1. Magnitude system
   (a) What is magnitude used in astronomy? Describe magnitude system.
   (b) What is Vega magnitude?
   (c) What is AB magnitude?
   (d) The magnitude of star A is m_A. The net flux of star A and B are I_A and I_B, respectively. What is the magnitude of star B m_B?
2. Photometric systems
   (a) List major photometric systems. Briefly describe each of them.
   (b) Describe pass-bands of filters used for SDSS photometric system.
3. Error propagation
   (a) What is error propagation?
   (b) Suppose the error for the quantity A is ΔA and the error for the quantity A is ΔB. What is the error of $A + B$?
   (c) Suppose the error for the quantity A is ΔA and the error for the quantity A is ΔB. What is the error of AB?
   (d) Suppose the flux of a star is S and the error of flux is N. Show that the error on magnitude is expressed as

$$\Delta m \sim 2.5 \log_{10} \left(1 + \frac{N}{S} \right).$$

4. Carry out aperture photometry of stars ID 7 and ID 13 on the image `lot_20210215_0116_df.fits`. Use the magnitude of the star ID 13 on the catalogue to calculate the magnitude of the star ID 7.

5. Pick two stars other than ID 7 and ID 13 on the image `lot_20210215_0117_df.fits`. Carry out aperture photometry of those two stars. Use the magnitude of one star shown in the catalogue to derive the magnitude of the other star. What is the magnitude and error of magnitude? Describe the method of your analysis and calculation. Show all the source codes of Python scripts you have written. Is your result consistent with the catalogue value?

6. Pick two stars other than ID 7 and ID 13 on the image `lot_20210215_0118_df.fits`. Carry out aperture photometry of those two stars. Use the magnitude of one star shown in the catalogue to derive the magnitude of the other star. What is the magnitude and error of magnitude? Describe the method of your analysis and calculation. Show all the source codes of Python scripts you have written. Is your result consistent with the catalogue value?

7. Pick two stars other than ID 7 and ID 13 on the image `lot_20210215_0206_df.fits`. Carry out aperture photometry of those two stars. Use the magnitude of one star shown in the catalogue to derive the magnitude of the other star. What is the magnitude and error of magnitude? Describe the method of your analysis and calculation. Show all the source codes of Python scripts you have written. Is your result consistent with the catalogue value?