Advanced Astronomical Observations 2021
Session 12: Aperture Photometry of Real Stars

Kinoshita Daisuke

28 April 2021
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 2021 to June 2021.
  - 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 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  54403      0  0:00:01  0:00:01 --:--:-- 54403
% ls -l
total 1
-rw-r--r--  1 daisuke  taiwan  65774 Apr 27 17:51 stds.tar.gz
```

Try following command to extract files.

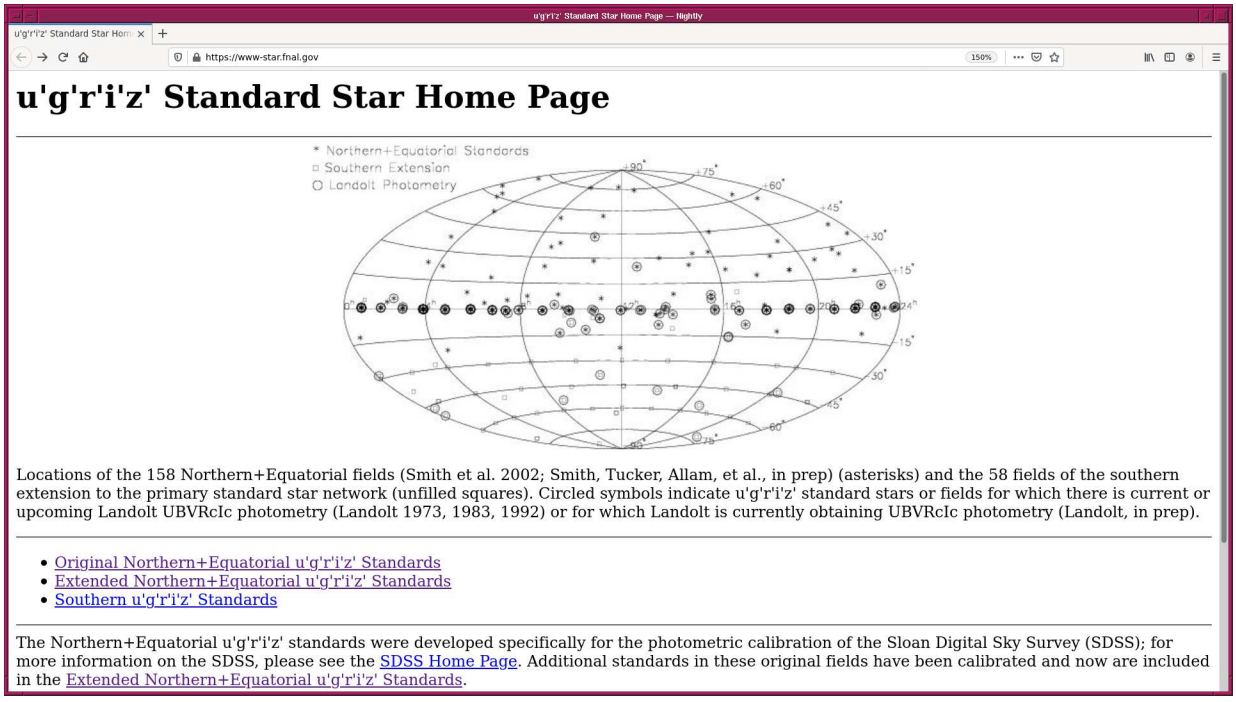

Figure 1: The u'g'r'i'z' Standard Star Home Page.

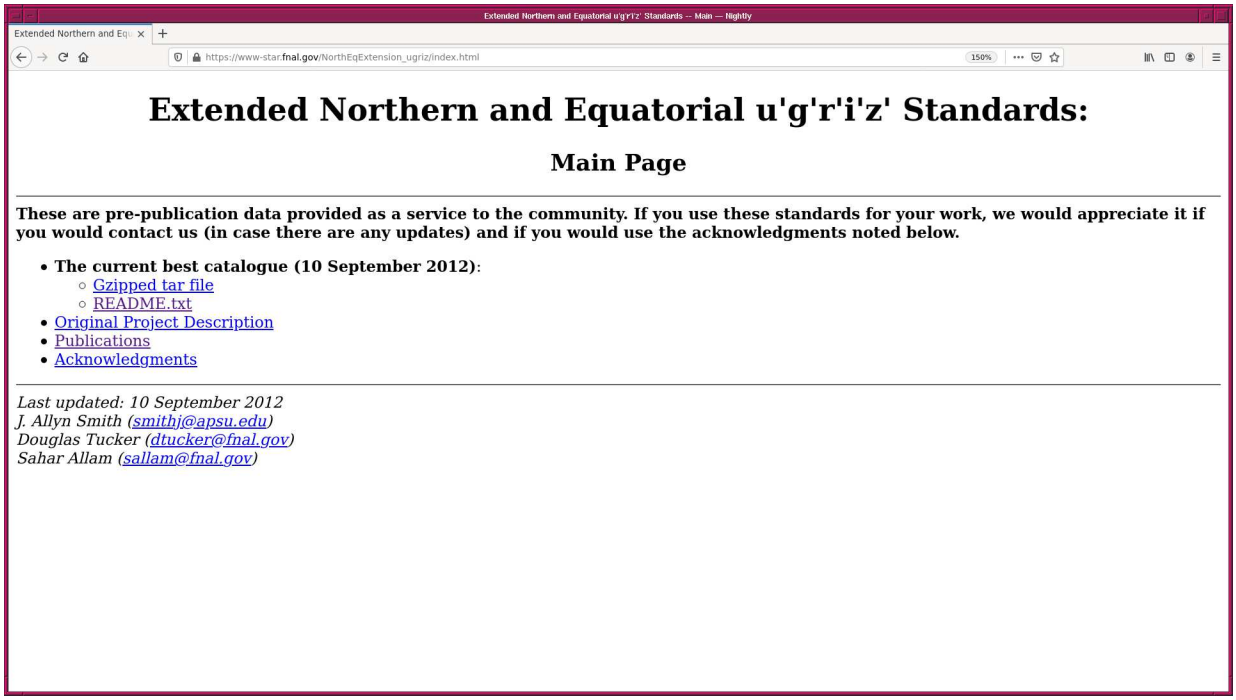

Figure 2: Extended Northern and Equatorial u'g'r'i'z' Standards web page.

```
% mkdir stds
% ls -l
total 1
drwxr-xr-x  2 daisuke  taiwan    512 Apr 27 18:58 stds/
-rw-r--r--  1 daisuke  taiwan  65774 Apr 27 17:51 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 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 -l
total 1
drwxr-xr-x  2 daisuke  taiwan   3584 Apr 27 18:59 stds/
-rw-r--r--  1 daisuke  taiwan  65774 Apr 27 17:51 stds.tar.gz
```

2 Coordinates of photometric standard stars

For this session, we process the image of standard star field PG1047+003. Make a Python script to read coordinates of stars in the field around the star PG1047+003.

Python Code 1: ao2021_s12_01.py

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

# importing astropy module
import astropy.coordinates

# data file name
file_data = 'stds/PG1047+003A.txt.clean.v3'

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

```
# 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 ao2021_s12_01.py
% ./ao2021_s12_01.py
# 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
```

3 Downloading SDSS image

Make a Python script to download SDSS image around the star PG1047+003.

Python Code 2: ao2021_s12_02.py

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

# importing argparse module
import argparse
```

```
# importing sys module
import sys

# importing astroquery module
import astroquery.simbad
import astroquery.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   = ['SDSSu', 'SDSSg', 'SDSSr', 'SDSSi', 'SDSSz']
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 ()

# checking output file name
if (file_output == ''):
    # printing error message
    print ("No output file name is given!")
    # exit
    sys.exit ()
elif not (file_output[-5:] == '.fits'):
    # printing error message
    print ("Output file must be FITS file!")
    # 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.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 ao2021_s12_02.py
% ./ao2021_s12_02.py -h
usage: ao2021_s12_02.py [-h] [-r {simbad,ned}]
                        [-s {SDSSu,SDSSg,SDSSr,SDSSi,SDSSz}] [-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}, --survey {SDSSu,SDSSg,SDSSr,SDSSi,SDSSz}
                        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

% ./ao2021_s12_02.py -r simbad -s SDSSr -t PG1047+003 -f 2048 -o pg1047_sdss.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/skv17620773117722.fits']
Downloading https://skyview.gsfc.nasa.gov/tempspace/fits/skv17620782820032.fits
|=========================================|  16M/ 16M (100.00%)        36s
% ls -l pg1047_sdss.fits
-rw-r--r--  1 daisuke  taiwan  16793280 Apr 27 21:23 pg1047_sdss.fits
```

4 Visualisation of SDSS image

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

```
% ginga pg1047_sdss.fits
```

5 Mark stars using aperture

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

Python Code 3: ao2021_s12_03.py

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

# importing argparse module
import argparse
```

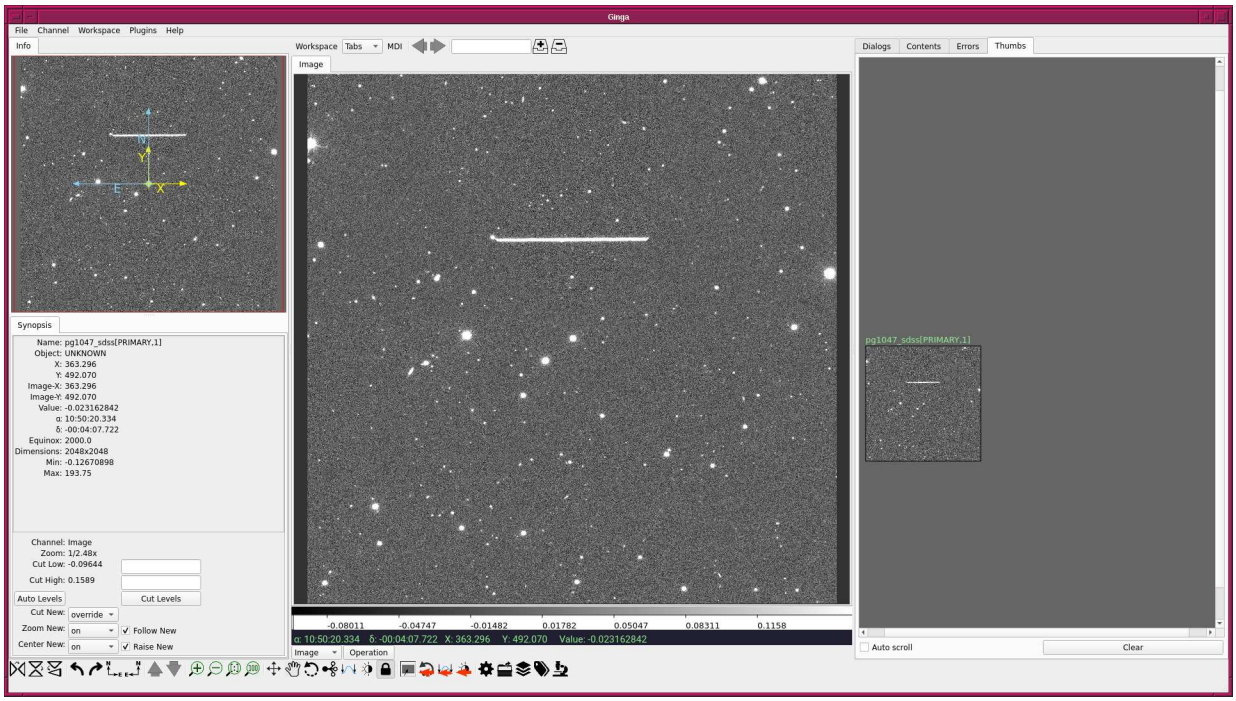

Figure 3: SDSS image around the star PG1047+003.

```
# importing sys module
import sys

# 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.pyplot

# constructing parser object
desc = 'Setting an aperture'
parser = argparse.ArgumentParser (description=desc)

# adding command-line arguments
parser.add_argument ('-r', '--radius', default=10.0, \
                     help='radius of aperture circle in arcsec')
parser.add_argument ('-s', '--std', default='', help='standard star file')
parser.add_argument ('-o', '--output', default='', \
                     help='output file name')
parser.add_argument ('file', nargs=1, default='', help='input file name')

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

# input parameters
```

```
radius_arcsec = args.radius
file_std      = args.std
file_output   = args.output
file_fits     = args.file[0]

# checking input file name
if (file_fits == ''):
    print ("You need to specify input file name.")
    sys.exit ()
if not (file_fits[-5:] == '.fits'):
    print ("Input file must be a FITS file.")
    sys.exit ()

# checking stdard star file
if (file_std == ''):
    print ("You need to specify input file name.")
    sys.exit ()

# checking output file name
if not ( (file_output[-4:] == '.eps') or (file_output[-4:] == '.pdf') \
         or (file_output[-4:] == '.png') or (file_output[-3:] == '.ps') ):
    print ("Output file must be either EPS, PDF, PNG, or PS.")
    sys.exit ()

# opening FITS file
hdu_list = astropy.io.fits.open (file_fits)

# reading FITS header
header = hdu_list[0].header

# WCS
wcs = astropy.wcs.WCS (header)

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

# closing FITS file
hdu_list.close ()

# making empty lists
list_ra  = []
list_dec = []

# 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])
        # appending RA and Dec to lists
        list_ra.append (ra_deg)
        list_dec.append (dec_deg)
```

```
# making skycoord object
positions = astropy.coordinates.SkyCoord (list_ra, list_dec, unit='deg')

# making aperture
u_arcsec = astropy.units.arcsec
aperture_std \
    = photutils.aperture.SkyCircularAperture (positions, \
                                              r=radius_arcsec * u_arcsec)
aperture_std_pix = aperture_std.to_pixel (wcs)

# making plot
matplotlib.pyplot.imshow (data, origin='lower', vmin=0, vmax=1)
aperture_std_pix.plot (color='red', lw=1.0)
matplotlib.pyplot.colorbar ()
matplotlib.pyplot.savefig (file_output, dpi=225)
```

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

```
% chmod a+x ao2021_s12_03.py
% ./ao2021_s12_03.py -h
usage: ao2021_s12_03.py [-h] [-r RADIUS] [-s STD] [-o OUTPUT] file

Setting an aperture

positional arguments:
  file                  input file name

optional arguments:
  -h, --help            show this help message and exit
  -r RADIUS, --radius RADIUS
                        radius of aperture circle in arcsec
  -s STD, --std STD     standard star file
  -o OUTPUT, --output OUTPUT
                        output file name

% ./ao2021_s12_03.py -s stds/PG1047+003A.txt.clean.v3 \
? -o pg1047_sdss.pdf pg1047_sdss.fits
% ls -l pg1047_sdss.pdf
-rw-r--r--  1 daisuke  taiwan  451335 Apr 27 22:05 pg1047_sdss.pdf
```

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

```
% xpdf pg1047_sdss.pdf
```

6 Choosing two stars

Pick two stars for your measurements. For example, following two stars are selected. (Fig. 5 and 6) The first star is at (α, δ) $\sim$ (10:50:02.8, -00:00:38), and the second star is at (α, δ) $\sim$ (10:50:07.9, -00:02:04). These two stars are the stars of ID 7 and 13.

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

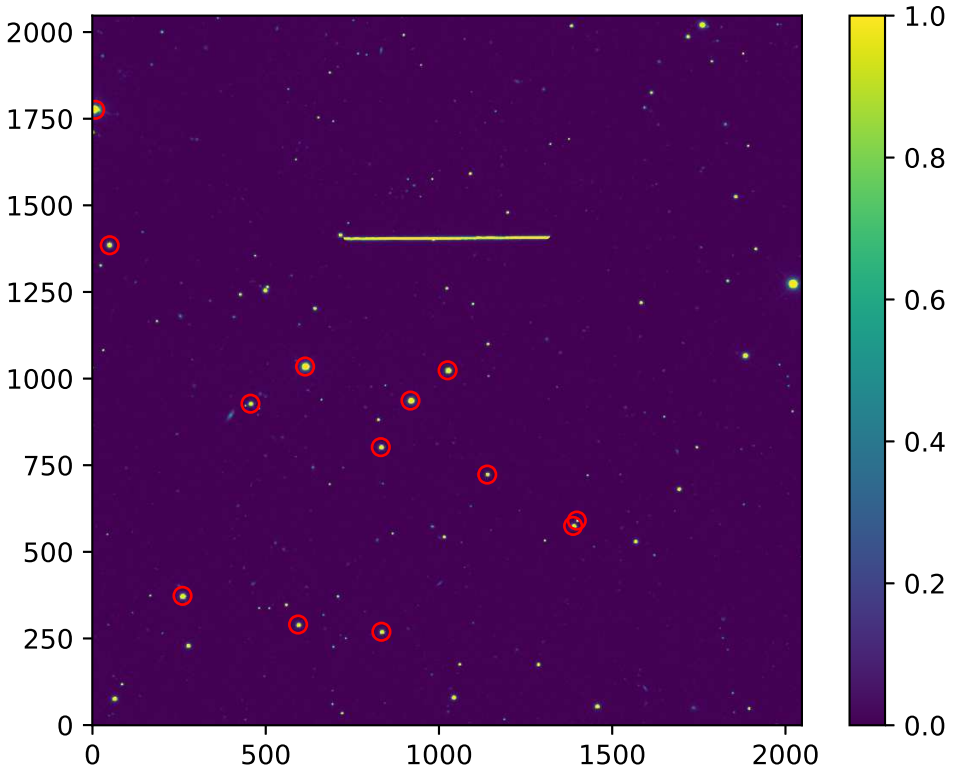

Figure 4: Locations of photometric standard stars.

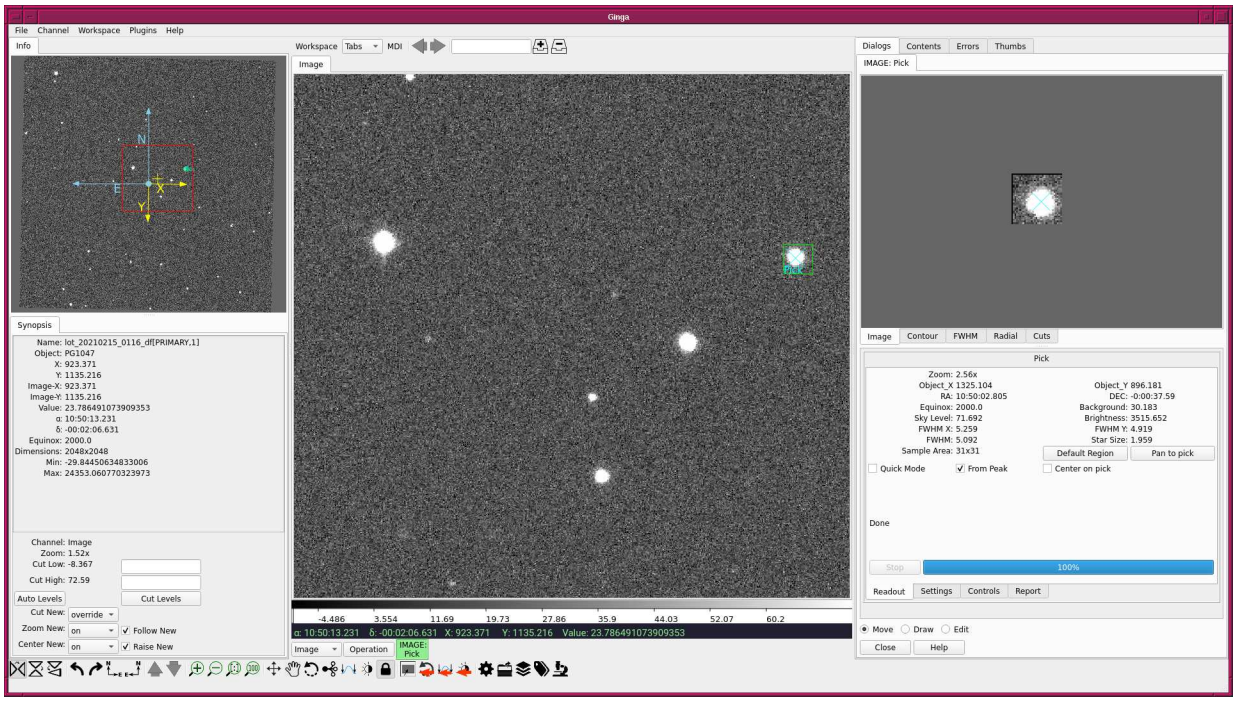

Figure 5: First star for the measurement.

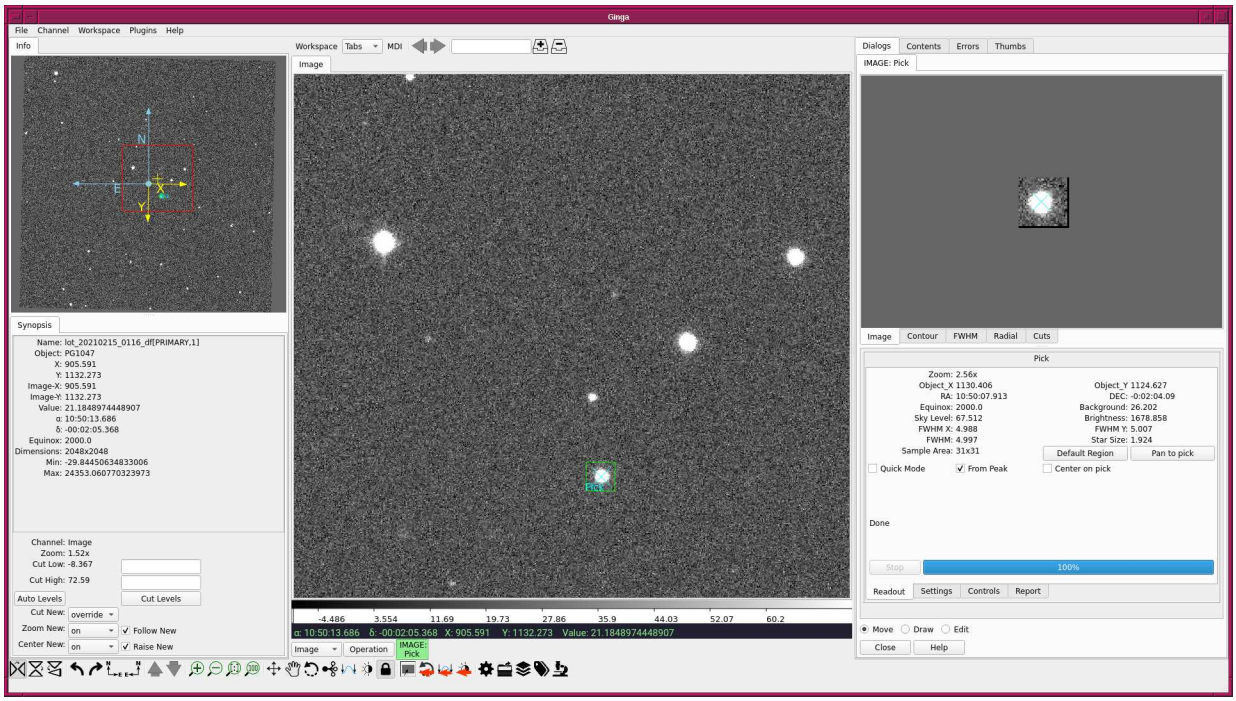

Figure 6: Second star for the measurement.

7 Identifying stars in our image

We have reduced the images taken on 15 February 2021 at the session 08. For this session, we use the image `lot_20210215_0116_df.fits`. Use Ginga to visualise the image. (Fig. 7)

```
% CAP -pi /somewhere/in/the/disk/lot_20210215_0116_df.fits .
% ls -l lot_20210215_0116_df.fits
-rw-r--r--  1 daisuke  taiwan  33566400 Apr  6 22:09 lot_20210215_0116_df.fits
% ginga 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. 8 and 9)

8 Measuring accurate position of stars using centroid

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

Python Code 4: ao2021_s12_04.py

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

# importing argparse module
import argparse

# importing sys module
import sys

# importing numpy module
import numpy

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

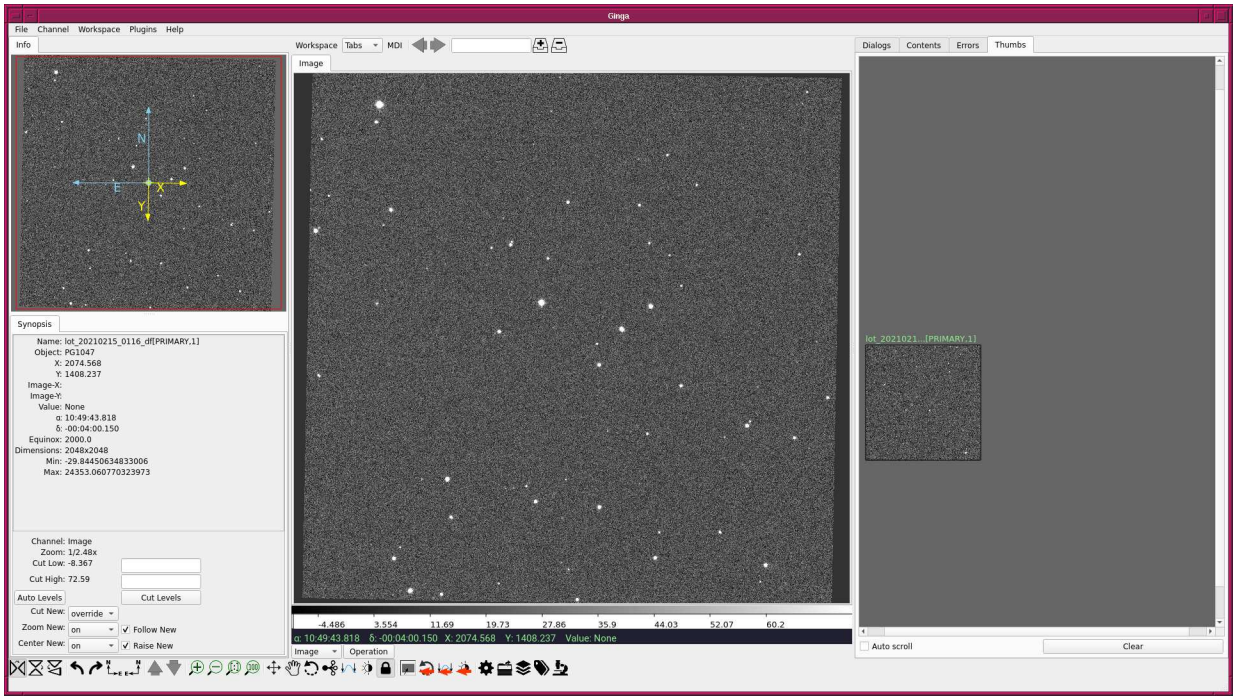

Figure 7: The image of `lot_20210215_0116_df.fits`.

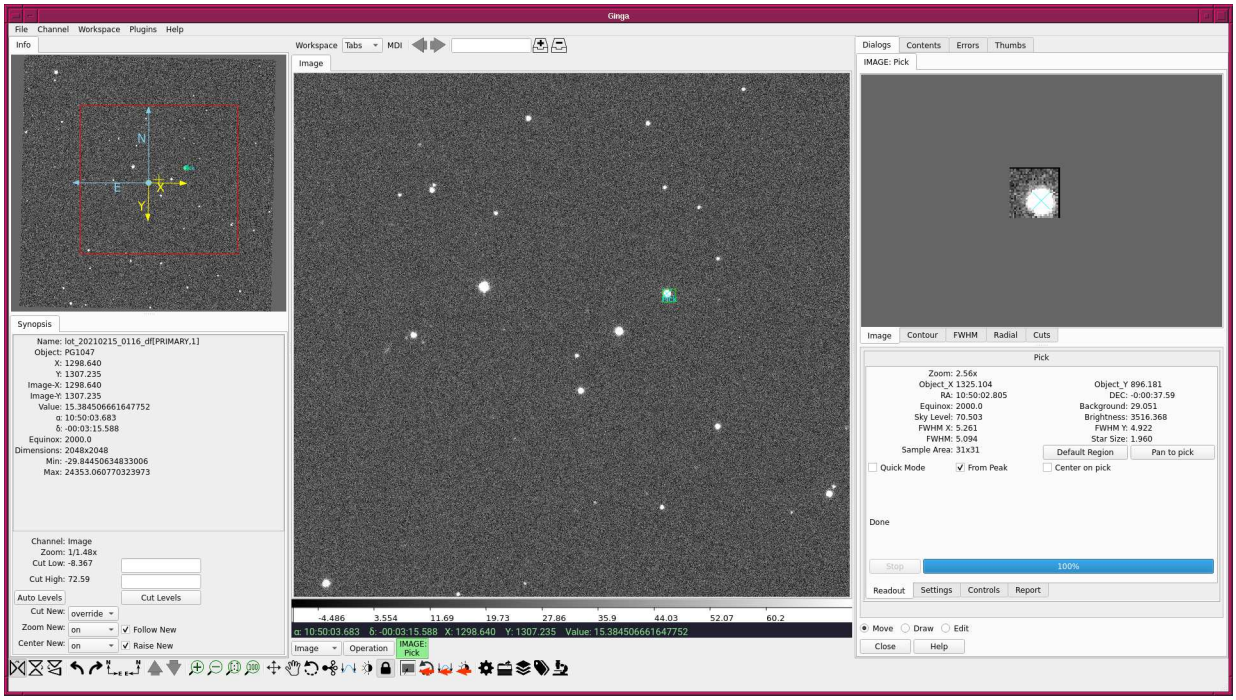

Figure 8: The location of star ID 7 on the image of `lot_20210215_0116_df.fits`.

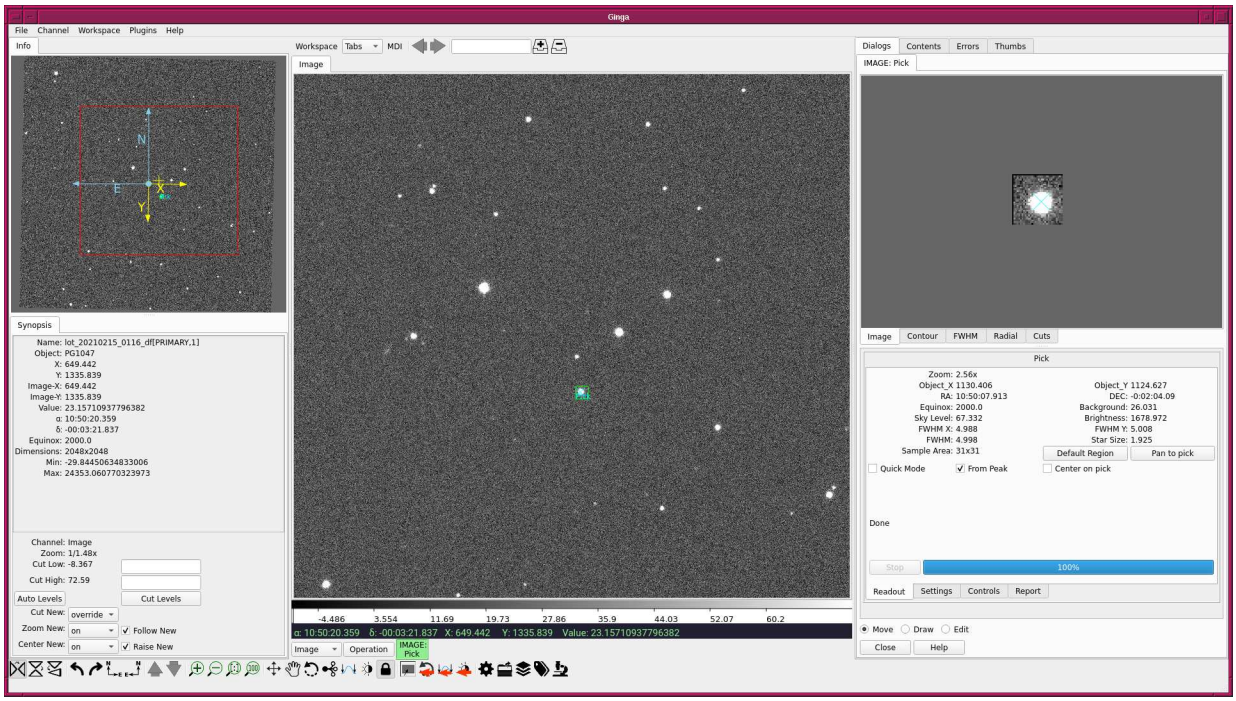

Figure 9: The location of star ID 13 on the image of `lot_20210215_0116_df.fits`.

```
import astropy.modeling

# importing photutils module
import photutils.centroids

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

# constructing parser object
desc = 'PSF fitting'
parser = argparse.ArgumentParser (description=desc)

# adding command-line arguments
list_psf = ['2dg', '2dm']
parser.add_argument ('-p', '--psf', choices=list_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 ('-o', '--output', default='psffitting.png', \
                     help='output file name')
parser.add_argument ('file', nargs=1, default='', help='input file name')

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

# input parameters
psf_model    = args.psf
half_width   = args.width
```

```
x_init      = args.xinit
y_init      = args.yinit
file_fits   = args.file[0]
file_output = args.output

# checking input file name
if (file_fits == ''):
    print ("You need to specify input file name.")
    sys.exit ()
if not (file_fits[-5:] == '.fits'):
    print ("Input file must be a FITS file.")
    sys.exit ()

# checking output file name
if not ( (file_output[-4:] == '.eps') or (file_output[-4:] == '.pdf') \
         or (file_output[-4:] == '.png') or (file_output[-3:] == '.ps') ):
    print ("Output file must be EPS or PDF or PNG or PS.")
    sys.exit ()

# opening FITS file
hdu_list = astropy.io.fits.open (file_fits)

# 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

# closing FITS file
hdu_list.close ()

# 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]

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

# centroid measurement
(xc_com, yc_com) = photutils.centroids.centroid_com (subframe)

# PSF fitting
```

```
subframe_y, subframe_x = numpy.indices (subframe.shape)
if (psf_model == '2dg'):
    psf_init = astropy.modeling.models.Gaussian2D (x_mean=xc_com, y_mean=yc_com)
elif (psf_model == '2dm'):
    psf_init = astropy.modeling.models.Moffat2D (x_0=xc_com, y_0=yc_com)
fit = astropy.modeling.fitting.LevMarLSQFitter ()
psf_fitted = fit (psf_init, subframe_x, subframe_y, subframe, maxiter=1000)

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

# printing information
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 ("#")
print ("# result")
print ("#  x_centre  = %f" % x_centre)
print ("#  y_centre  = %f" % y_centre)
print ("#  amplitude = %f" % amplitude)
if (psf_model == '2dg'):
    print ("#  theta     = %f" % theta)
    print ("#  x_fwhm    = %f" % x_fwhm)
    print ("#  y_fwhm    = %f" % y_fwhm)
elif (psf_model == '2dm'):
    print ("#  alpha     = %f" % alpha)
    print ("#  gamma     = %f" % gamma)
    print ("#  fwhm      = %f" % fwhm)
print ("#")

# printing result
print ("# X_CENTRE, Y_CENTRE, FWHM")
print ("%f %f %f" % (x_centre, y_centre, fwhm) )

# 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 - %d [pixel]" % x_min)
ax.set_ylabel ("Y - %d [pixel]" % y_min)

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

# saving file
fig.savefig (file_output, dpi=225)
```

Run the script.

```
% chmod a+x ao2021_s12_04.py
% ./ao2021_s12_04.py -h
usage: ao2021_s12_04.py [-h] [-p {2dg,2dm}] [-w WIDTH] [-x XINIT] [-y YINIT]
                        [-o OUTPUT]
                        file

PSF fitting

positional arguments:
  file                  input file name

optional arguments:
  -h, --help            show this help message and exit
  -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
  -o OUTPUT, --output OUTPUT
                        output file name

% ./ao2021_s12_04.py -p 2dg -w 35 -x 1325 -y 896 \
? -o star1_centroid.pdf lot_20210215_0116_df.fits
#
# input file name = lot_20210215_0116_df.fits
# half-width of search box = 35.000000
# x_init = 1325.000000
# y_init = 896.000000
#
# result
#  x_centre  = 1323.966923
#  y_centre  = 895.113398
#  amplitude = 3432.805245
#  theta     = 10.900625
#  x_fwhm    = 5.068980
#  y_fwhm    = 5.366122
#
# X_CENTRE, Y_CENTRE, FWHM
1323.966923 895.113398 5.217551
```

The star ID 7 is located at (x, y) = (1323.97, 895.11) and its FWHM is 5.22 pixel. (Fig. 10)
Next, measure the position and FWHM of star ID 13.

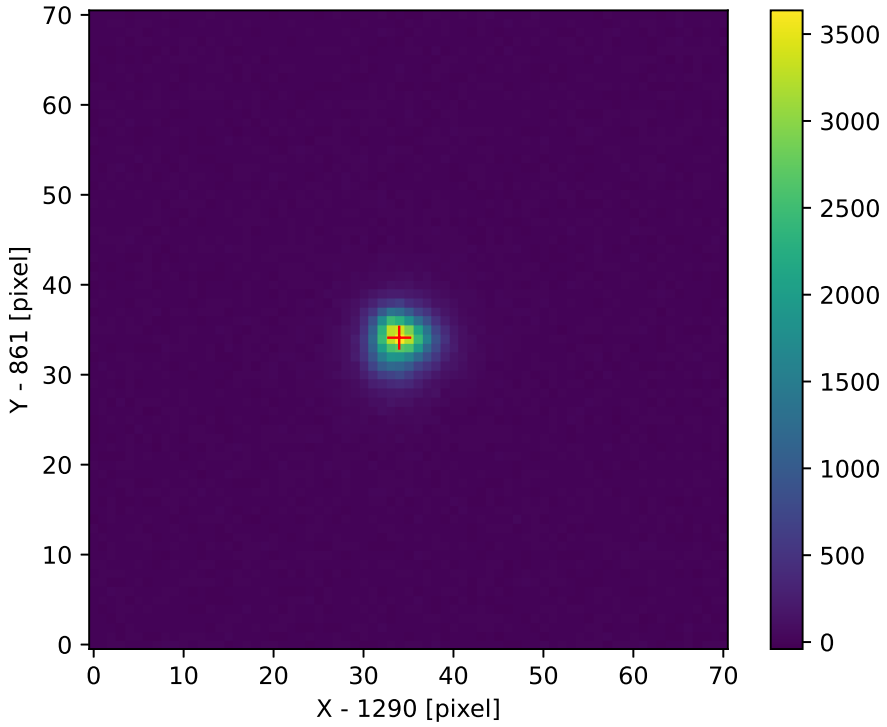

Figure 10: The centroid measurement of the star ID 7.

```
% ./ao2021_s12_04.py -p 2dg -w 35 -x 1130 -y 1125 \
? -o star2_centroid.pdf lot_20210215_0116_df.fits
#
# input file name = lot_20210215_0116_df.fits
# half-width of search box = 35.000000
# x_init = 1130.000000
# y_init = 1125.000000
#
# result
#  x_centre  = 1129.322355
#  y_centre  = 1123.505494
#  amplitude = 1639.085545
#  theta     = -6.004077
#  x_fwhm    = 5.239266
#  y_fwhm    = 5.087628
#
# X_CENTRE, Y_CENTRE, FWHM
1129.322355 1123.505494 5.163447
```

The star ID 13 is located at (x, y) = (1129.32, 1123.51) and its FWHM is 5.16 pixel. (Fig. 11)

9 Aperture photometry of two stars

Carry out aperture photometry of two stars.

Python Code 5: ao2021_s12_05.py

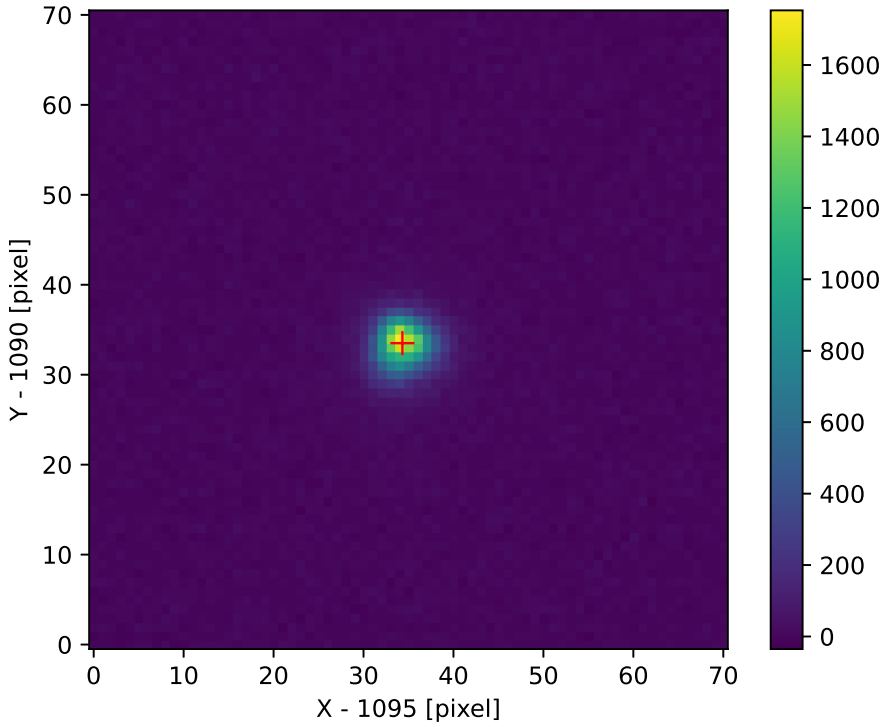

Figure 11: The centroid measurement of the star ID 13.

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

# importing argparse module
import argparse

# importing sys module
import sys

# importing numpy module
import numpy

# importing astropy module
import astropy.io.fits
import astropy.modeling
import astropy.stats

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

# importing matplotlib module
import matplotlib.pyplot

# constructing parser object
desc = 'sky background estimate using sigma-clipping algorithm'
parser = argparse.ArgumentParser (description=desc)

# 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 ('-o', '--output', default='', \
                     help='output file name')
parser.add_argument ('file', nargs=1, 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
file_output           = args.output
file_fits             = args.file[0]

# aperture radii 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

# checking input file name
if (file_fits == ''):
    print ("You need to specify input file name.")
    sys.exit ()
if not (file_fits[-5:] == '.fits'):
    print ("Input file must be a FITS file.")
    sys.exit ()

# checking output file name
if not ( (file_output[-4:] == '.eps') or (file_output[-4:] == '.pdf') \
         or (file_output[-4:] == '.png') or (file_output[-3:] == '.ps') ):
    print ("Output file must be either EPS, PDF, PNG, or PS.")
    sys.exit ()

# opening FITS file
hdu_list = astropy.io.fits.open (file_fits)

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

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

# closing FITS file
hdu_list.close ()

# 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

# position of the centre
position = (x_centre, y_centre)

# making apertures (circular aperture for star and circular annulus for sky)
apphot_aperture \
    = photutils.aperture.CircularAperture (position, r=aperture_radius_pixel)
apphot_annulus \
    = photutils.aperture.CircularAnnulus (position, \
                                          r_in=skyannulus_inner_pixel, \
                                          r_out=skyannulus_outer_pixel)

# making masked data for sky annulus
skyannulus_data       = apphot_annulus.to_mask (method='center').multiply (data)
skyannulus_mask       = skyannulus_data <= 0.0
skyannulus_maskeddata = numpy.ma.array (skyannulus_data, mask=skyannulus_mask)

# sky background estimate using sigma-clipping algorithm
skybg_mean, skybg_median, skybg_stddev \
    = astropy.stats.sigma_clipped_stats (skyannulus_maskeddata, \
                                         sigma=3.0, maxiters=10, \
                                         cenfunc='median')

# sky background
skybg_per_pixel = skybg_median

# aperture photometry
noise = numpy.sqrt (data)
phot_star = photutils.aperture.aperture_photometry (data, apphot_aperture, \
                                                    error=noise)

# net flux
net_flux = phot_star['aperture_sum'] - skybg_per_pixel * apphot_aperture.area

# error of net flux
net_flux_error = phot_star['aperture_sum_err']

# printing result of aperture photometry
```

```
print ("net flux = %f +/- %f" % (net_flux, net_flux_error) )

# making masked data for sky annulus
skyannulus_data       = apphot_annulus.to_mask (method='center').multiply (data)
skyannulus_mask       = skyannulus_data <= 0.0
skyannulus_maskeddata = numpy.ma.array (skyannulus_data, mask=skyannulus_mask)

# making plot
matplotlib.pyplot.xlabel ("X [pixel]")
matplotlib.pyplot.ylabel ("Y [pixel]")
matplotlib.pyplot.xlim (x_min, x_max)
matplotlib.pyplot.ylim (y_min, y_max)
matplotlib.pyplot.imshow (data, origin='lower', vmin=5000, vmax=10000)
matplotlib.pyplot.plot (x_centre, y_centre, marker='+', \
                        color='red', markersize=10)
apphot_aperture.plot (color='yellow', lw=2.0)
apphot_annulus.plot (color='green', lw=2.0)
matplotlib.pyplot.colorbar ()
matplotlib.pyplot.savefig (file_output, dpi=225)
```

Execute the script, and measure the brightness of the star ID 7.

```
% chmod a+x ao2021_s12_05.py
% ./ao2021_s12_05.py -h
usage: ao2021_s12_05.py [-h] [-f FWHM] [-a APERTURE] [-s1 SKYANNULUS1]
                        [-s2 SKYANNULUS2] [-w WIDTH] [-x XCENTRE] [-y YCENTRE]
                        [-o OUTPUT]
                        file

sky background estimate using sigma-clipping algorithm

positional arguments:
  file                  input file name

optional arguments:
  -h, --help            show this help message and exit
  -f FWHM, --fwhm FWHM  FWHM of stellar PSF in pixel (default: 4.0)
  -a APERTURE, --aperture APERTURE
                        aperture radius in FWHM (default: 1.5)
  -s1 SKYANNULUS1, --skyannulus1 SKYANNULUS1
                        inner sky annulus radius in FWHM (default: 3.0)
  -s2 SKYANNULUS2, --skyannulus2 SKYANNULUS2
                        outer sky annulus radius in FWHM (default: 5.0)
  -w WIDTH, --width WIDTH
                        half-width of subframe to be plotted (default: 15)
  -x XCENTRE, --xcentre XCENTRE
                        x coordinate of target
  -y YCENTRE, --ycentre YCENTRE
                        y coordinate of target
  -o OUTPUT, --output OUTPUT
                        output file name

% ./ao2021_s12_05.py -f 5.22 -a 1.5 -s1 3.0 -s2 5.0 -w 35 -x 1323.97 -y 895.11 \
? -o star1_aperture.pdf lot_20210215_0116_df.fits
/home/daisuke/tex/astro/NCU/Lecture/AdvObs_2020b/12_apphot2/script_11/./ao2021_s
12_05.py:136: RuntimeWarning: invalid value encountered in sqrt
  noise = numpy.sqrt (data)
net flux = 112194.196930 +/- 340.580641
```

The net flux of star ID 7 is 112194 ± 341 ADU.
Next, measure the brightness of star ID 13.

```
% ./ao2021_s12_05.py -f 5.16 -a 1.5 -s1 3.0 -s2 5.0 -w 35 -x 1129.32 -y 1123.51 \
? -o star2_aperture.pdf lot_20210215_0116_df.fits
/home/daisuke/tex/astro/NCU/Lecture/AdvObs_2020b/12_apphot2/script_11/./ao2021_s
12_05.py:136: RuntimeWarning: invalid value encountered in sqrt
  noise = numpy.sqrt (data)
net flux = 52276.536341 +/- 236.335462
```

The net flux of star ID 13 is 52277 ± 236 ADU.
Apertures and sky annuli for stars ID 7 and ID 13 are shown in Fig. 12 and 13.

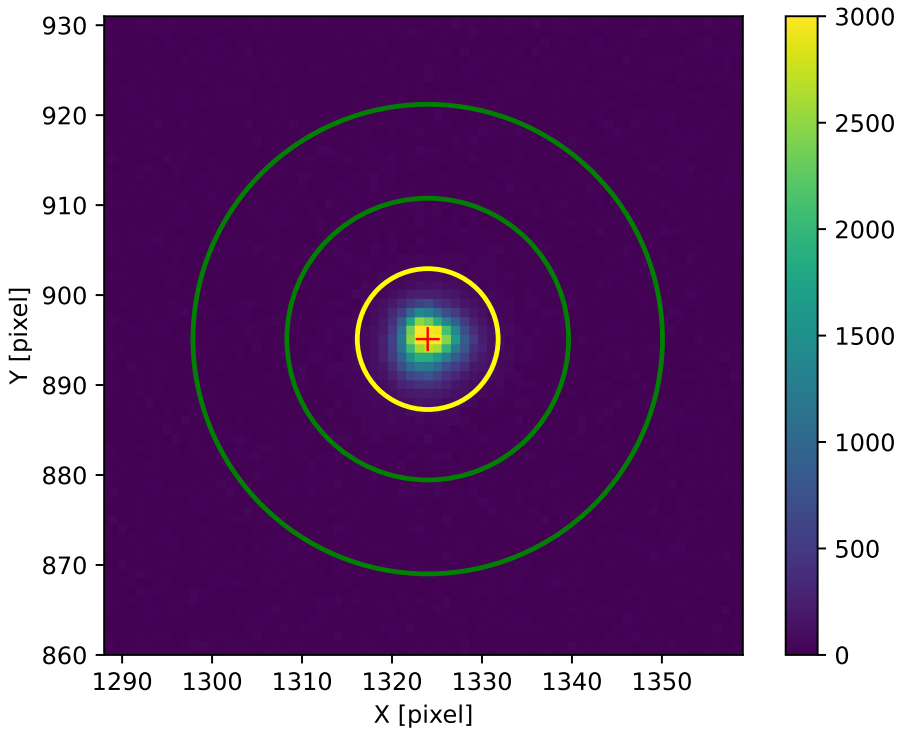

Figure 12: Aperture and sky annulus set for star ID 7.

10 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 112194 ± 341 ADU and 52277 ± 236 ADU, respectively. Use these information to calculate the magnitude of star ID 13.

Python Code 6: ao2021_s12_06.py

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

# importing math module
import math

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

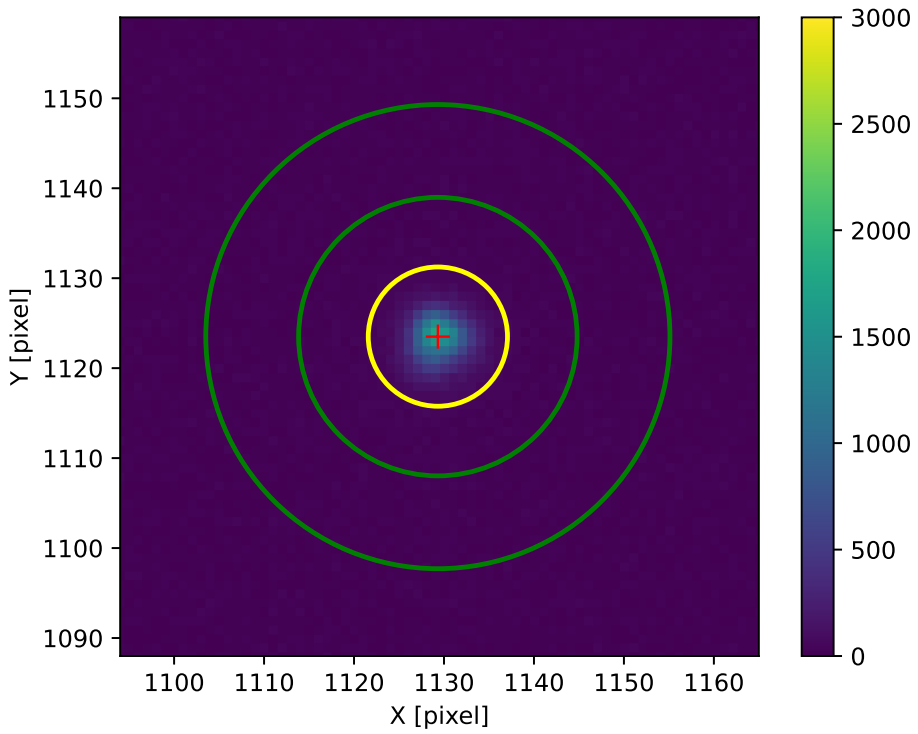

Figure 13: Aperture and sky annulus set for star ID 13.

```
# net flux of star ID 7
flux_star1 = 112194.0

# flux error of star ID 7
err_star1 = 341.0

# net flux of star ID 13
flux_star2 = 52277.0

# flux error of star ID 13
err_star2 = 236.0

# 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 ao2021_s12_06.py
% ./ao2021_s12_06.py
#
# input parameters
#
#  mag_star1  = 13.718000
#  flux_star1 = 112194.000000 ADU
#  err_star1  = 341.000000 ADU
#  flux_star2 = 52277.000000 ADU
#  err_star2  = 236.000000 ADU
#
mag_star2 = 14.547147 +/- 0.005897
```

The magnitude of star ID 13 is calculated to be 14.547 ± 0.006. 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.

11 For your training

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`

12 Assignment

1. What is magnitude used in astronomy? Describe magnitude system.
2. What is Vega magnitude?
3. What is AB magnitude?
4. List major photometric systems. Briefly describe each of them.
5. Describe pass-bands of filters used for SDSS photometric system.
6. 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?
7. What is error propagation?
8. Suppose the error for the quantity A is ΔA and the error for the quantity A is ΔB. What is the error of $A + B$?

9. Suppose the error for the quantity A is ΔA and the error for the quantity A is ΔB. What is the error of AB?

10. Suppose the flux of a star is S and the error of flux is N. Show that the error on magnitude is expressed as

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

11. 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.

12. 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?

13. 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?

14. 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?