Advanced Astronomical Observations 2021
Session 11: Aperture Photometry

Kinoshita Daisuke

23 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/`

Aperture photometry is a simple and effective way to measure the brightness of point-like sources on astronomical images. For this session, we try aperture photometry using `photutils` package.

1 Photutils package

For this session, `photutils` package is needed. Visit following websites to learn about `photutils`.

- photutils
  - `https://photutils.readthedocs.io/` (Fig. 1)
  - `https://pypi.org/project/photutils/`
  - `https://github.com/astropy/photutils`

Try following to check whether you have `photutils` package installed on your computer. If you can import `photutils` package successfully, then you have `photutils` package properly installed on your computer.

```
% python3.9
Python 3.9.2 (default, Feb 21 2021, 12:39:42)
[GCC 7.5.0] on netbsd9
Type "help", "copyright", "credits" or "license" for more information.
>>> import photutils
>>>
```

If you see an error message like below, then you do not have `photutils` package on your computer.

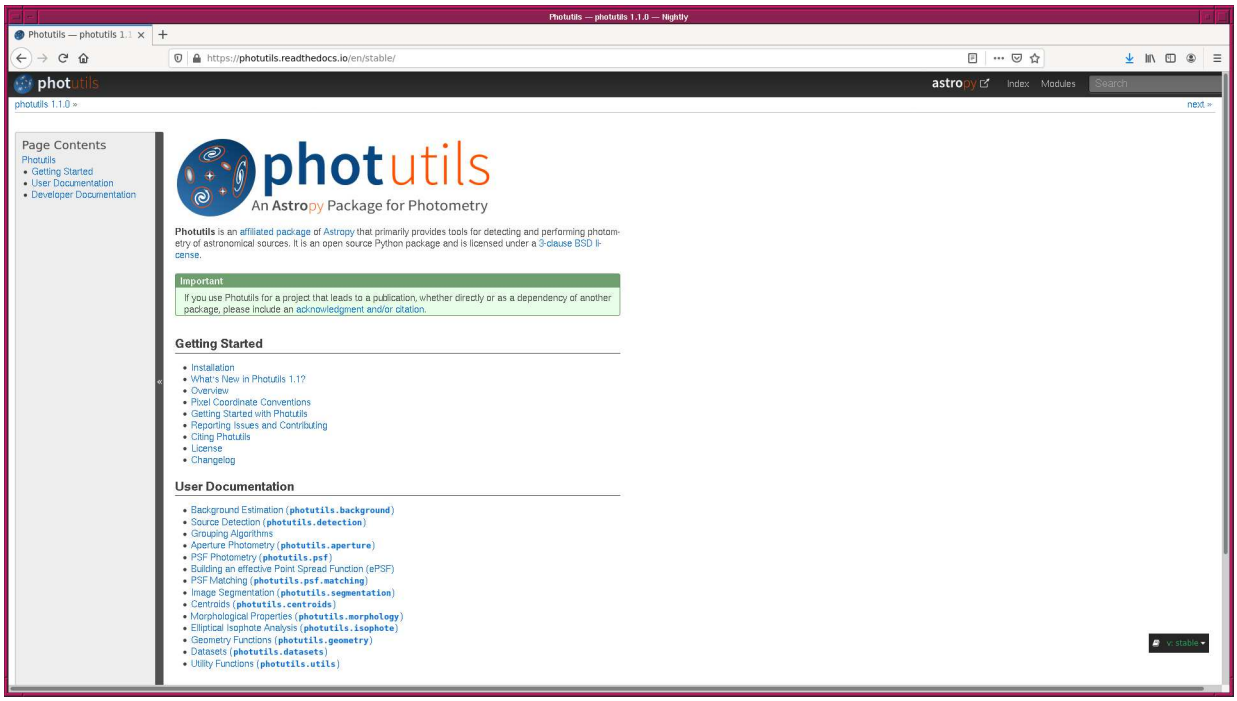

Figure 1: The official website of `photutils` package at `https://photutils.readthedocs.io/`.

```
% python3.9
Python 3.9.2 (default, Mar 27 2021, 17:26:25)
[GCC 7.5.0] on netbsd9
Type "help", "copyright", "credits" or "license" for more information.
>>> import photutils
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ModuleNotFoundError: No module named 'photutils'
>>>
```

If you do not have `photutils` pakcage, visit the official website of `photutils` package (Fig. 2), read the installation instruction, and install the package on your computer.

2 Generating synthetic images

Use `photutils.datasets` to generate a synthetic image for aperture photometry.

2.1 Making a synthetic image for aperture photometry

Make a Python script to generate an image simulating stars.

Python Code 1: ao2021_s11_01.py

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

# importing argparse module
import argparse

# importing sys module
import sys
```

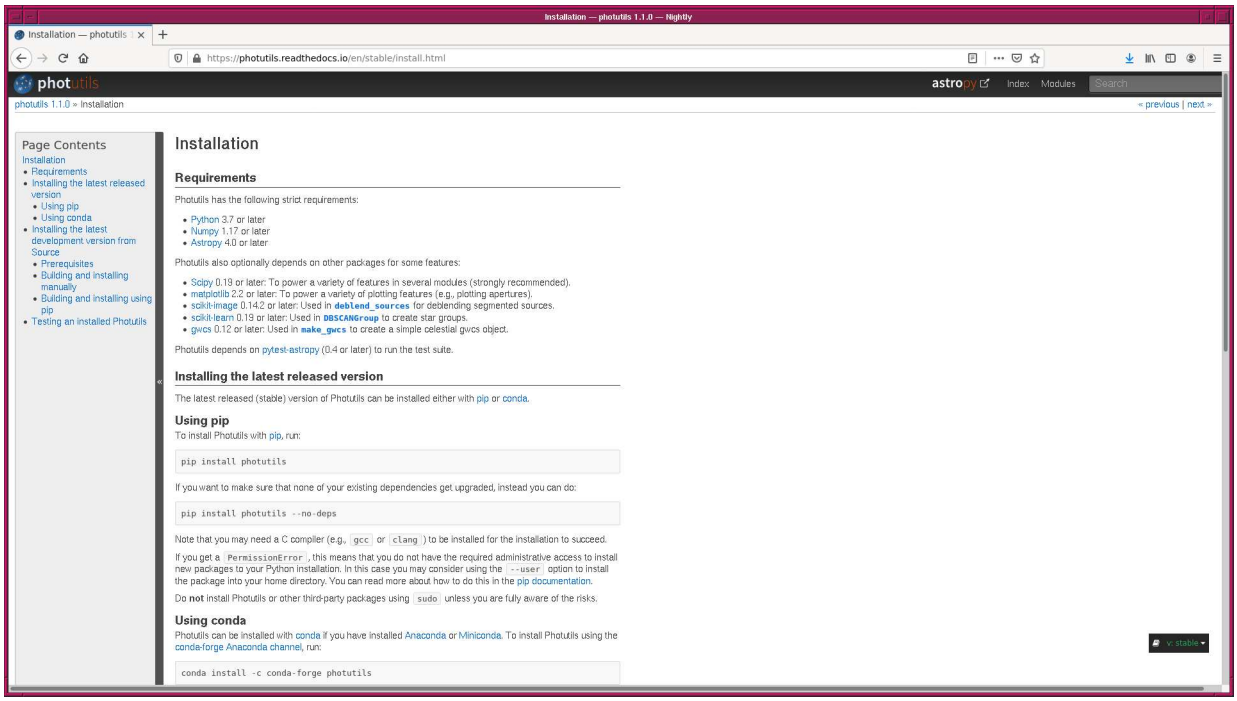

Figure 2: The installation instruction and description of the requirements at the official website of `photutils` package at `https://photutils.readthedocs.io/en/stable/install.html`.

```
# importing numpy module
import numpy

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

# importing photutils module
import photutils.datasets

# constructing parser object
desc = 'generating a synthetic image'
parser = argparse.ArgumentParser (description=desc)

# adding command-line arguments
parser.add_argument ('-b', '--background', type=float, default=2500.0, \
                     help='background level (default: 2500)')
parser.add_argument ('-s', '--sigma', type=float, default=50.0, \
                     help='noise level (default: 50)')
parser.add_argument ('-n', '--nstar', type=int, default=5, \
                     help='number of stars to generate (default: 5)')
parser.add_argument ('-f1', '--fluxmin', type=float, default=100000.0, \
                     help='min total flux of star (default: 100000)')
parser.add_argument ('-f2', '--fluxmax', type=float, default=200000.0, \
                     help='max total flux of star (default: 200000)')
parser.add_argument ('-p1', '--psfmin', type=float, default=4.0, \
                     help='min FWHM of stellar radial profile (default: 4)')
parser.add_argument ('-p2', '--psfmax', type=float, default=6.0, \
                     help='max FWHM of stellar radial profile (default: 6)')
parser.add_argument ('-o', '--output', default='', \
                     help='output file name')
```

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

# input parameters
sky_background_level = args.background
noise_level          = args.sigma
nstar                = args.nstar
flux_total_min       = args.fluxmin
flux_total_max       = args.fluxmax
psf_fwhm_min         = args.psfmin
psf_fwhm_max         = args.psfmax
file_output          = args.output

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

# image size
image_size_x = 1024
image_size_y = 1024
image_size   = (image_size_x, image_size_y)
coord_min = 100.0
coord_max = 900.0

# grid of data
#
# x coord ==> [100, 300, 500, 700, 900, 100, 300, 500, 700, 900, 100, ...]
# y coord ==> [100, 100, 100, 100, 100, 300, 300, 300, 300, 300, 500, ...]
#
list_x    = numpy.array ([])
list_y    = numpy.array ([])
list_flux = numpy.array ([])
list_psf  = numpy.array ([])
for i in range (nstar):
    list_x \
        = numpy.append (list_x, numpy.linspace (coord_min, coord_max, nstar) )
for y in numpy.linspace (coord_min, coord_max, nstar):
    list_y = numpy.append (list_y, numpy.repeat (y, nstar) )
for i in range (nstar):
    list_flux \
        = numpy.append (list_flux, \
                        numpy.linspace (flux_total_min, flux_total_max, nstar) )
for psf in numpy.linspace (psf_fwhm_min, psf_fwhm_max, nstar):
    list_psf = numpy.append (list_psf, numpy.repeat (psf, nstar) )

# generating sky background
image_background \
    = photutils.datasets.make_noise_image (image_size, \
                                           distribution='gaussian', \
                                           mean=sky_background_level, \
                                           stddev=noise_level)

# making a source table
fwhm_sigma = 2.0 * numpy.sqrt (2.0 * numpy.log (2.0) )
source_table = astropy.table.Table ()
```

```
source_table['x_0']    = list_x
source_table['y_0']    = list_y
source_table['flux']   = list_flux
source_table['sigma'] = list_psf / fwhm_sigma

# printing source table
print ("source_table:")
print (source_table)

# generating stars
image_star \
    = photutils.datasets.make_gaussian_prf_sources_image (image_size, \
                                                          source_table)

# making synthetic image by adding background and stars
image = image_background + image_star

# writing a FITS file
print ("Now, writing data into FITS file \"%s\"..." % file_output)
astropy.io.fits.writeto (file_output, image)
print ("Finished writing data!")
```

Execute the script and generate a synthetic image.

```
% chmod a+x ao2021_s11_01.py
% ./ao2021_s11_01.py -h
./ao2021_s11_01.py -h
usage: ao2021_s11_01.py [-h] [-b BACKGROUND] [-s SIGMA] [-n NSTAR]
                        [-f1 FLUXMIN] [-f2 FLUXMAX] [-p1 PSFMIN] [-p2 PSFMAX]
                        [-o OUTPUT]

generating a synthetic image

optional arguments:
  -h, --help            show this help message and exit
  -b BACKGROUND, --background BACKGROUND
                        background level (default: 2500)
  -s SIGMA, --sigma SIGMA
                        noise level (default: 50)
  -n NSTAR, --nstar NSTAR
                        number of stars to generate (default: 5)
  -f1 FLUXMIN, --fluxmin FLUXMIN
                        min total flux of star (default: 100000)
  -f2 FLUXMAX, --fluxmax FLUXMAX
                        max total flux of star (default: 200000)
  -p1 PSFMIN, --psfmin PSFMIN
                        min FWHM of stellar radial profile (default: 4)
  -p2 PSFMAX, --psfmax PSFMAX
                        max FWHM of stellar radial profile (default: 6)
  -o OUTPUT, --output OUTPUT
                        output file name

% ./ao2021_s11_01.py -b 6400 -s 80 -p1 3 -p2 7 -o synstars_0.fits
source_table:
 x_0   y_0    flux          sigma
----- ----- -------- ------------------
100.0 100.0 100000.0 1.2739827004320285
300.0 100.0 125000.0 1.2739827004320285
```

```
500.0 100.0 150000.0 1.2739827004320285
700.0 100.0 175000.0 1.2739827004320285
900.0 100.0 200000.0 1.2739827004320285
100.0 300.0 100000.0 1.6986436005760381
300.0 300.0 125000.0 1.6986436005760381
500.0 300.0 150000.0 1.6986436005760381
700.0 300.0 175000.0 1.6986436005760381
900.0 300.0 200000.0 1.6986436005760381
100.0 500.0 100000.0 2.1233045007200477
300.0 500.0 125000.0 2.1233045007200477
500.0 500.0 150000.0 2.1233045007200477
700.0 500.0 175000.0 2.1233045007200477
900.0 500.0 200000.0 2.1233045007200477
100.0 700.0 100000.0  2.547965400864057
300.0 700.0 125000.0  2.547965400864057
500.0 700.0 150000.0  2.547965400864057
700.0 700.0 175000.0  2.547965400864057
900.0 700.0 200000.0  2.547965400864057
100.0 900.0 100000.0  2.972626301008067
300.0 900.0 125000.0  2.972626301008067
500.0 900.0 150000.0  2.972626301008067
700.0 900.0 175000.0  2.972626301008067
900.0 900.0 200000.0  2.972626301008067
Now, writing data into FITS file "synstars_0.fits"...
Finished writing data!
% ls -l synstars_0.fits
-rw-r--r--  1 daisuke  taiwan  8392320 Apr 22 19:11 synstars_0.fits
```

2.2 Examining pixel values

Make a Python script to examine pixel values of generated synthetic image.

Python Code 2: ao2021_s11_02.py

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

# importing argparse module
import argparse

# importing numpy module
import numpy
import numpy.ma

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

# construction pf parser object
desc = 'calculating statistical information of FITS files'
parser = argparse.ArgumentParser (description=desc)

# adding arguments
list_rejection = ['NONE', 'sigclip']
parser.add_argument ('-r', '--rejection', default='NONE', \
                     choices=list_rejection, \
                     help='rejection algorithm (default: NONE)')
parser.add_argument ('-t', '--threshold', type=float, default=4.0, \
                     help='threshold for sigma clipping (default: 4.0)')
parser.add_argument ('-n', '--maxiters', type=int, default=10, \
```

```
                    help='maximum number of iterations (default: 10)')
parser.add_argument ('files', nargs='+', help='FITS files')

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

# input parameters
rejection  = args.rejection
threshold  = args.threshold
maxiters   = args.maxiters
list_files = args.files

# printing information
print ("# Input parameters")
print ("#   rejection algorithm = %s" % rejection)
if not (rejection == 'NONE'):
    print ("#   threshold of sigma-clipping = %f" % threshold)
    print ("#   maximum number of iterations = %d" % maxiters)

# printing header
print ("#")
print ("# %-25s %8s %8s %8s %7s %8s %8s" \
       % ('file name', 'npix', 'mean', 'median', 'stddev', 'min', 'max') )
print ("#")

# scanning files
for file_fits in list_files:
    # if the file is not a FITS file, then skip
    if not (file_fits[-5:] == '.fits'):
        continue

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

    # header of primary HDU
    header = hdu_list[0].header

    # closing FITS file
    hdu_list.close ()

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

    # image of primary HDU
    data0 = hdu_list[0].data

    # closing FITS file
    hdu_list.close ()

    # calculations

    # for no rejection algorithm
    if (rejection == 'NONE'):
        # making a masked array
        data1 = numpy.ma.array (data0, mask=False)
    # for sigma clipping algorithm
    elif (rejection == 'sigclip'):
        data1 = numpy.ma.array (data0, mask=False)
        # iterations
```

```
        for j in range (maxiters):
            # number of usable pixels of previous iterations
            npix_prev = len (numpy.ma.compressed (data1) )
            # calculation of median
            median = numpy.ma.median (data1)
            # calculation of standard deviation
            stddev = numpy.ma.std (data1)
            # lower threshold
            low = median - threshold * stddev
            # higher threshold
            high = median + threshold * stddev
            # making a mask
            mask = (data1 < low) | (data1 > high)
            # making a masked array
            data1 = numpy.ma.array (data0, mask=mask)
            # number of usable pixels
            npix_now = len (numpy.ma.compressed (data1) )
            # leaving the loop, if number of usable pixels do not change
            if (npix_now == npix_prev):
                break

    # calculation of mean, median, stddev, min, and max
    mean   = numpy.ma.mean (data1)
    median = numpy.ma.median (data1)
    stddev = numpy.ma.std (data1)
    vmin   = numpy.ma.min (data1)
    vmax   = numpy.ma.max (data1)

    # number of pixels
    npix = len (data1.compressed () )

    # file name
    filename = file_fits.split ('/') [-1]

    # printing result
    print ("%-27s %8d %8.2f %8.2f %7.2f %8.2f %8.2f" \
           % (filename, npix, mean, median, stddev, vmin, vmax) )
```

Run the script and check pixel values.

```
% chmod a+x ao2021_s11_02.py
% ./ao2021_s11_02.py -h
usage: ao2021_s11_02.py [-h] [-r {NONE,sigclip}] [-t THRESHOLD] [-n MAXITERS]
                        files [files ...]

calculating statistical information of FITS files

positional arguments:
  files                 FITS files

optional arguments:
  -h, --help            show this help message and exit
  -r {NONE,sigclip}, --rejection {NONE,sigclip}
                        rejection algorithm (default: NONE)
  -t THRESHOLD, --threshold THRESHOLD
                        threshold for sigma clipping (default: 4.0)
  -n MAXITERS, --maxiters MAXITERS
                        maximum number of iterations (default: 10)
```

```
% ./ao2021_s11_02.py synstars_0.fits
# Input parameters
#   rejection algorithm = NONE
#
# file name                      npix      mean    median   stddev       min       max
#
synstars_0.fits                1048576   6403.60   6400.36   137.93   6016.47 25014.00
```

The mean pixel value is $\sim$ 6400 ADU as expected.

2.3 Visual inspection of the image

Use Ginga to show the image. 25 stars are seen on the image. (Fig. 3)

```
% ginga synstars_0.fits
```

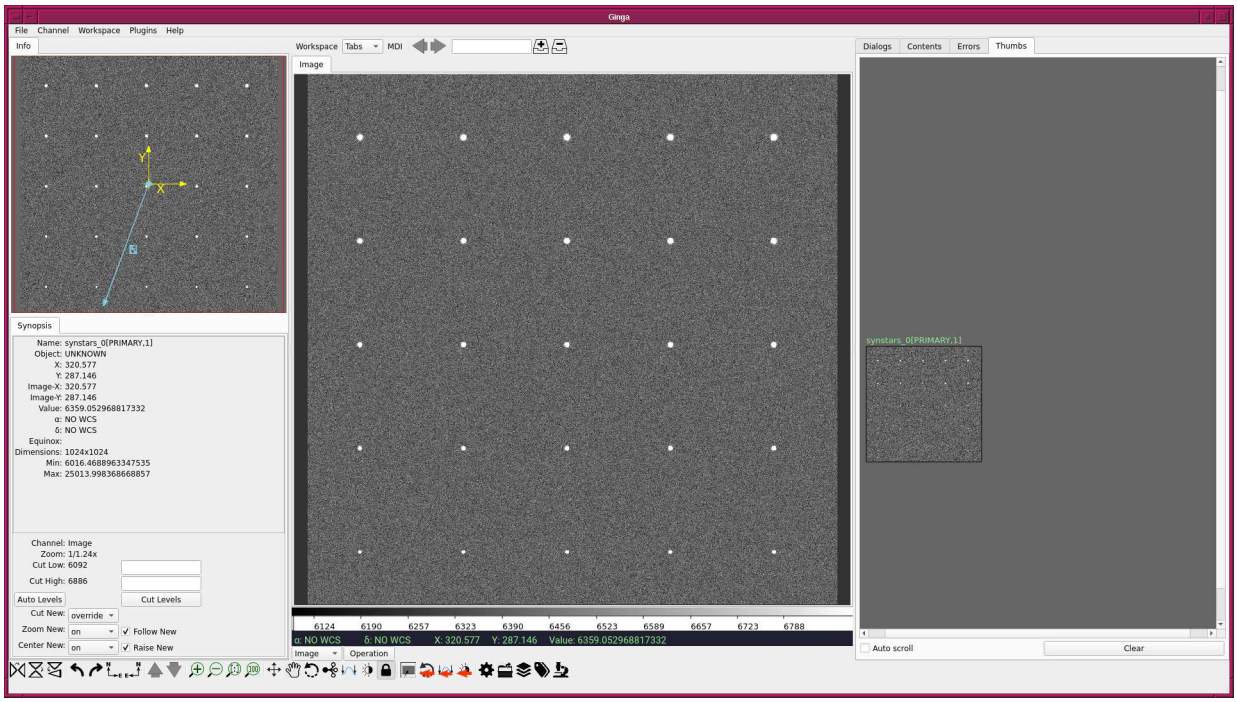

Figure 3: The synthetic image generated by photutils.datasets module.

3 Centroid measurements

3.1 Knowing a rough position of a star using Ginga

Use Ginga to know a rough position of a star.

1. Start Ginga.
2. Load the image "`synstars_0.fits`".
3. Choose a star for your centroid measurement.
4. Zoom in and pan to the star you have picked.
5. Visit the menu "Plugins".

6. Choose “Analysis, and then click the menu “Pick”.
7. Move your mouse cursor to your target.
8. Click the left button of your mouse.
9. Read values of (x, y) coordinate of your target.

For example, a star at the centre of the image gives the coordinate (x, y) $\sim$ (501, 501). (Fig. 4)

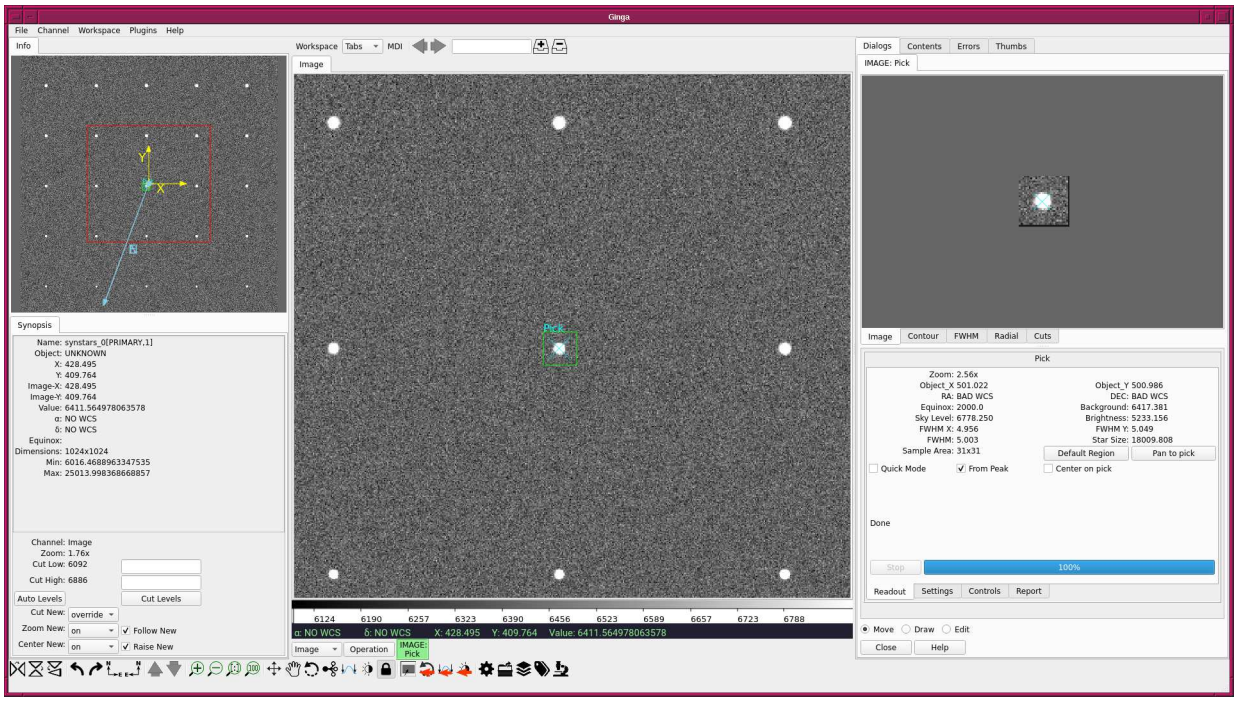

Figure 4: The position of a star is read using Ginga.

Actually, the coordinate shown by IRAF, SAOimage DS9, Ginga, etc. are different from the coordinate used by Python/Numpy. The coordinate used by IRAF, SAOimage DS9, Ginga, etc. starts from 1, but the coordinate used by Python/Numpy starts from 0. Therefore, The coordinate we actually need is (x, y) $\sim$ (500, 500). But, don’t worry, it’s usually no problem using the value shown by Ginga directly. For more information, read following document.

- `https://photutils.readthedocs.io/en/stable/pixel_conventions.html`

3.2 Measuring the centre of the star

Use 2-dimensional Gaussian function to find the centre of the star.

Python Code 3: ao2021_s11_03.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
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 and measure the coordinate of the centre of star.

```
% chmod a+x ao2021_s11_03.py
% ./ao2021_s11_03.py -h
usage: ao2021_s11_03.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_s11_03.py -w 35 -x 501 -y 501 -o centroid_0.png synstars_0.fits
#
# input file name = synstars_0.fits
# half-width of search box = 35.000000
# x_init = 501.000000
# y_init = 501.000000
#
# result
#  x_centre  = 500.002516
#  y_centre  = 499.991020
#  amplitude = 5220.195173
#  theta     = -2.804108
#  x_fwhm    = 5.008186
#  y_fwhm    = 5.049099
#
# X_CENTRE, Y_CENTRE, FWHM
500.002516 499.991020 5.028643
% ls -l centroid_0.png
-rw-r--r--  1 daisuke  taiwan  71930 Apr 22 20:38 centroid_0.png
```

The coordinate of the centre of star is measured to be $(x, y) = (500.00, 499.99)$. The FWHM of stellar PSF is measured to be 5.03 pixel.

Show the plot produced to check the result of centroid measurement. (Fig. 5)

```
% feh -dF centroid_0.png
```

4 Aperture photometry

Now, we know the position of a star you picked. We can start aperture photometry to measure the brightness of the star.

4.1 Setting an aperture for the star

First, we set an aperture for the star. For this session, radius of the aperture is set to be twice of FWHM. Make a Python script to set an aperture and make a plot around the star.

Python Code 4: ao2021_s11_04.py

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

# importing argparse module
import argparse
```

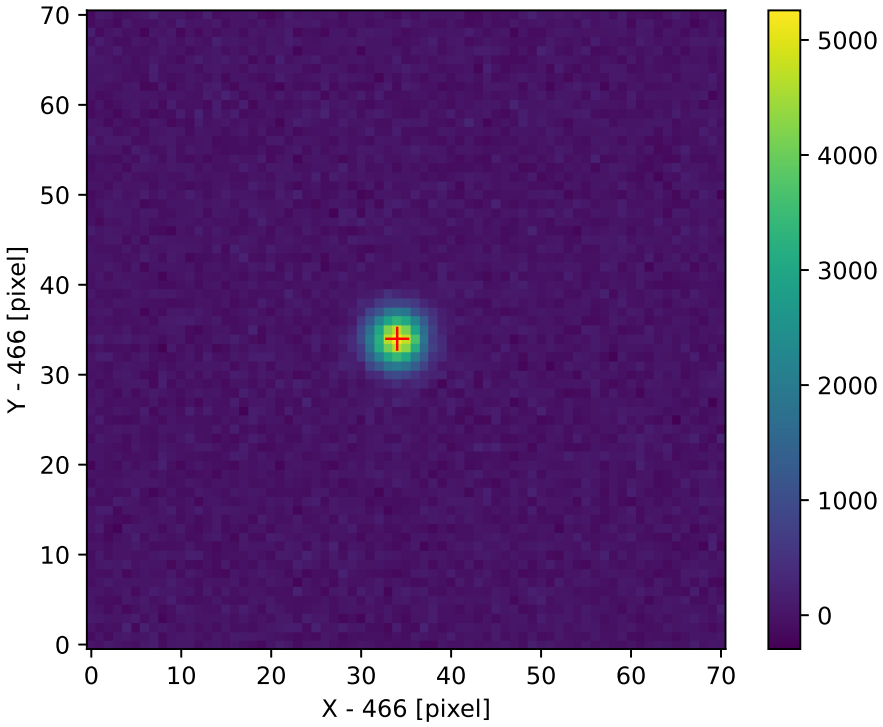

Figure 5: The result of centroid measurement using 2-dimensional Gaussian function.

```
# importing sys module
import sys

# importing numpy module
import numpy

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

# importing photutils module
import photutils.centroids
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 ('-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 ('-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
half_width           = args.width
x_centre             = args.xcentre
y_centre             = args.ycentre
file_output          = args.output
file_fits            = args.file[0]

# aperture radius in pixel
aperture_radius_pixel = aperture_radius_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
subframe = data[y_min:y_max, x_min:x_max]

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

# printing aperture
print (apphot_aperture)

# making plot
matplotlib.pyplot.xlabel ("X - %d [pixel]" % x_min)
matplotlib.pyplot.ylabel ("Y - %d [pixel]" % y_min)
matplotlib.pyplot.imshow (subframe, origin='lower')
matplotlib.pyplot.plot (x_centre_sub, y_centre_sub, marker='+', \
                        color='red', markersize=10)
apphot_aperture.plot (color='yellow', lw=2.0)
matplotlib.pyplot.savefig (file_output, dpi=225)
```

Execute the script.

```
% chmod a+x ao2021_s11_04.py
% ./ao2021_s11_04.py -f 5.03 -a 2.0 -w 35 -x 500.00 -y 499.99 \
? -o aperture_0.png synstars_0.fits
Aperture: CircularAperture
positions: [35.  , 35.99]
r: 10.06
% ls -l aperture_0.png
-rw-r--r--  1 daisuke  taiwan  67086 Apr 22 20:47 aperture_0.png
```

Show the plot. (Fig. 6)

```
% feh -dF aperture_0.png
```

4.2 Setting a sky annulus

Second, set a sky annulus for sky background estimation.

Python Code 5: ao2021_s11_05.py

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

# importing argparse module
import argparse
```

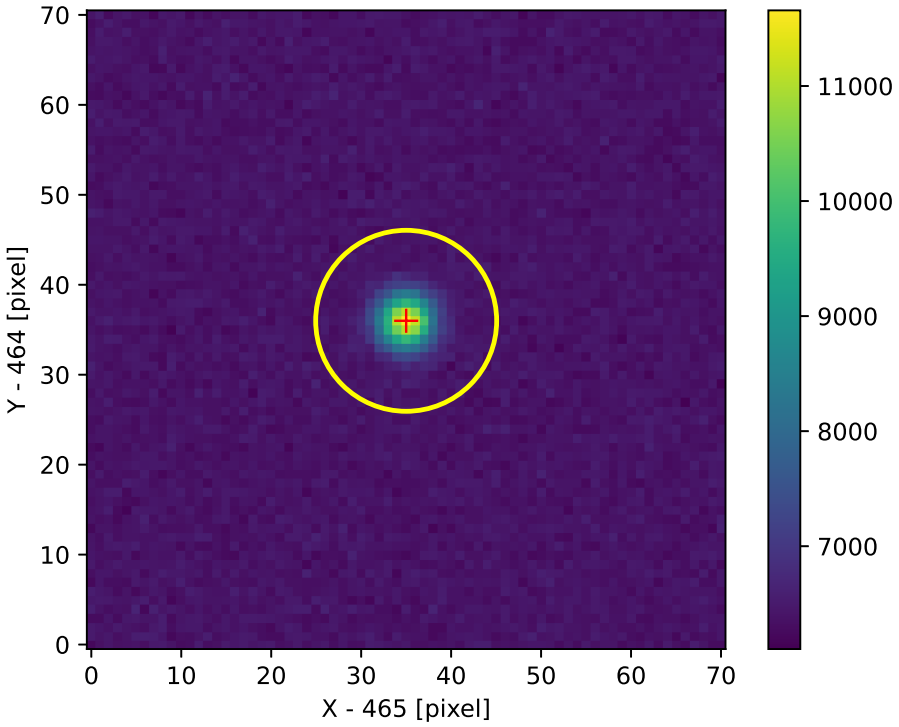

Figure 6: The location of the aperture set for the star.

```
# importing sys module
import sys

# importing numpy module
import numpy

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

# importing photutils module
import photutils.centroids
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 ('-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
subframe = data[y_min:y_max, x_min:x_max]

# 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 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)

# printing aperture
print (apphot_aperture)
print (apphot_annulus)

# making plot
matplotlib.pyplot.xlabel ("X - %d [pixel]" % x_min)
matplotlib.pyplot.ylabel ("Y - %d [pixel]" % y_min)
matplotlib.pyplot.imshow (subframe, origin='lower')
matplotlib.pyplot.plot (x_centre_sub, y_centre_sub, marker='+', \
                        color='red', markersize=10)
apphot_aperture.plot (color='yellow', lw=2.0)
apphot_annulus.plot (color='green', lw=2.0)
matplotlib.pyplot.savefig (file_output, dpi=225)
```

Run the script. For here, we set the inner radius of sky annulus to be 3 times of FWHM and the outer radius of sky annulus to be 6 times of FWHM.

```
% chmod a+x ao2021_s11_05.py
% ./ao2021_s11_05.py -h
usage: ao2021_s11_05.py [-h] [-f FWHM] [-a APERTURE] [-s1 SKYANNULUS1]
                        [-s2 SKYANNULUS2] [-w WIDTH] [-x XCENTRE] [-y YCENTRE]
                        [-o OUTPUT]
                        file

Setting an aperture

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_s11_05.py -f 5.03 -a 2.0 -s1 3.0 -s2 6.0 -w 35 -x 500.00 -y 499.99 \
? -o aperture_1.png synstars_0.fits
Aperture: CircularAperture
positions: [35.  , 35.99]
r: 10.06
Aperture: CircularAnnulus
positions: [35.  , 35.99]
r_in: 15.09
r_out: 30.18
% ls -l aperture_1.png
-rw-r--r--  1 daisuke  taiwan  113903 Apr 22 21:16 aperture_1.png
```

Show the plot to visualise the locations of aperture and sky annulus. (Fig. 7)

4.3 Adding pixel values within the aperture

Third, we add pixel values within the aperture. The function `aperture_photometry ()` can be used for this.

Python Code 6: ao2021_s11_06.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
import astropy.modeling

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

# importing matplotlib module
```

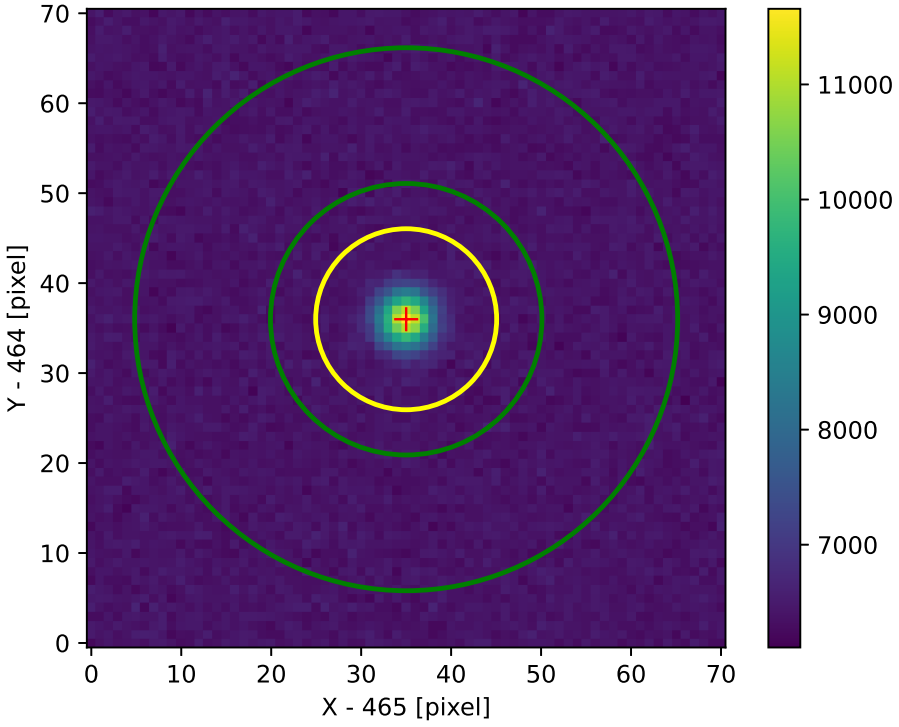

Figure 7: The location of the aperture and sky annulus set for the star. The circular aperture for the star is shown by yellow circle, and the circular sky annulus for sky background estimate is shown by green circle.

```
import matplotlib.pyplot

# constructing parser object
desc = 'measuring brightness of star + sky'
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 ('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_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 ()

# 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
subframe = data[y_min:y_max, x_min:x_max]

# 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 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)

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

# printing result
print (apphot_star)
print ("aperture sum       = %f ADU" % apphot_star['aperture_sum'])
print ("aperture sum error = %f ADU" % apphot_star['aperture_sum_err'])
```

Run the script.

```
% chmod a+x ao2021_s11_06.py
% ./ao2021_s11_06.py -h
usage: ao2021_s11_06.py [-h] [-f FWHM] [-a APERTURE] [-s1 SKYANNULUS1]
                        [-s2 SKYANNULUS2] [-w WIDTH] [-x XCENTRE] [-y YCENTRE]
                        file

measuring brightness of star + sky

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

% ./ao2021_s11_06.py -f 5.03 -a 2.0 -s1 3.0 -s2 6.0 -w 35 -x 500.00 -y 499.99 \
? synstars_0.fits
 id xcenter      ycenter         aperture_sum     aperture_sum_err
      pix           pix
--- ------- ----------------- ------------------ -----------------
  1    35.0 35.99000000000001 2186481.1466926844 1478.675470376338
aperture sum       = 2186481.146693 ADU
aperture sum error = 1478.675470 ADU
```

The total flux within the aperture is measured to be 2.19×10^6 ADU. Note that the flux we have measured is the flux of combined signal of the star and sky background. To know the net flux of the star, we have to subtract sky background component.

4.4 A simple way to estimate sky background level

A very simple way to estimate sky background level is to calculate the mean value of pixels within the sky annulus. Here is a Python script to do it.

Python Code 7: ao2021_s11_07.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
import astropy.modeling

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

# importing matplotlib module
import matplotlib.pyplot

# constructing parser object
```

```
desc = 'measuring brightness of star + sky'
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 ('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_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 ()

# 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
subframe = data[y_min:y_max, x_min:x_max]

# 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 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)

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

# sky background estimate
apphot_sky \
    = photutils.aperture.aperture_photometry (subframe, apphot_annulus, \
                                              error=noise)
skybg_per_pixel = apphot_sky['aperture_sum'] / apphot_annulus.area

# printing result
print (apphot_star)
print ("aperture sum       = %f ADU" % apphot_star['aperture_sum'])
print ("aperture sum error = %f ADU" % apphot_star['aperture_sum_err'])
print (apphot_sky)
print ("sky background per pixel       = %f ADU" % skybg_per_pixel)
```

Execute the script and measure the sky background level.

```
% chmod a+x ao2021_s11_07.py
% ./ao2021_s11_07.py -h
usage: ao2021_s11_07.py [-h] [-f FWHM] [-a APERTURE] [-s1 SKYANNULUS1]
                        [-s2 SKYANNULUS2] [-w WIDTH] [-x XCENTRE] [-y YCENTRE]
                        file
```

```
measuring brightness of star + sky

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

% ./ao2021_s11_07.py -f 5.03 -a 2.0 -s1 3.0 -s2 6.0 -w 35 -x 500.00 -y 499.99 \
? synstars_0.fits
 id xcenter      ycenter         aperture_sum     aperture_sum_err
      pix           pix
--- ------- ----------------- ------------------ -----------------
  1    35.0 35.99000000000001 2186481.1466926844 1478.675470376338
aperture sum       = 2186481.146693 ADU
aperture sum error = 1478.675470 ADU
 id xcenter      ycenter         aperture_sum     aperture_sum_err
      pix           pix
--- ------- ----------------- ------------------ ------------------
  1    35.0 35.99000000000001 13727128.615859833 3705.0139832205537
sky background per pixel       = 6396.318719 ADU
```

Measured sky background level is 6396.3 ADU which is consistent with the input parameter given to the Python script for synthetic image generation.

4.5 Result of sky background subtraction

Subtract the sky background from the sum of pixel values within the aperture.

Python Code 8: ao2021_s11_08.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
import astropy.modeling
```

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

# importing matplotlib module
import matplotlib.pyplot

# constructing parser object
desc = 'measuring brightness of star by sky subtraction'
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 ('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_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 ()

# 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
subframe = data[y_min:y_max, x_min:x_max]

# 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 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)

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

# sky background estimate
apphot_sky \
    = photutils.aperture.aperture_photometry (subframe, apphot_annulus, \
                                              error=noise)
skybg_per_pixel = apphot_sky['aperture_sum'] / apphot_annulus.area

# sky background subtraction
#   net flux = total flux within aperture
#               - skybg per pixel * number of pixels in aperture
net_flux = apphot_star['aperture_sum'] - skybg_per_pixel * apphot_aperture.area
net_flux_error = apphot_star['aperture_sum_err']

# printing result
print (apphot_star)
```

```
print ("aperture sum       = %f ADU" % apphot_star['aperture_sum'])
print ("aperture sum error = %f ADU" % apphot_star['aperture_sum_err'])
print (apphot_sky)
print ("sky background per pixel       = %f ADU" % skybg_per_pixel)
print ("")
print ("net flux of star = %f +/- %f ADU" % (net_flux, net_flux_error) )
```

Run the script, and check the result.

```
% chmod a+x ao2021_s11_08.py
% ./ao2021_s11_08.py -h
usage: ao2021_s11_08.py [-h] [-f FWHM] [-a APERTURE] [-s1 SKYANNULUS1]
                        [-s2 SKYANNULUS2] [-w WIDTH] [-x XCENTRE] [-y YCENTRE]
                        file

measuring brightness of star by sky subtraction

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

% ./ao2021_s11_08.py -f 5.03 -a 2.0 -s1 3.0 -s2 6.0 -w 35 -x 500.00 -y 499.99 \
? synstars_0.fits
 id xcenter      ycenter         aperture_sum     aperture_sum_err
      pix           pix
--- ------- ----------------- ------------------ -----------------
  1    35.0 35.99000000000001 2186481.1466926844 1478.675470376338
aperture sum       = 2186481.146693 ADU
aperture sum error = 1478.675470 ADU
 id xcenter      ycenter         aperture_sum     aperture_sum_err
      pix           pix
--- ------- ----------------- ------------------ ------------------
  1    35.0 35.99000000000001 13727128.615859833 3705.0139832205537
sky background per pixel       = 6396.318719 ADU

net flux of star = 152832.462862 +/- 1478.675470 ADU
```

Estimated net flux of the star is 153,000 ADU and it is roughly consistent with what we expect (150,000 ADU).

5 Aperture photometry with better sky background estimate

A simple mean of pixel values within the sky annulus may not give an accurate estimate for the sky background level if there are some stars within the sky annulus. A better way is to use sigma-clipping algorithm for sky background

estimation.

5.1 Extraction of pixel values within sky annulus

Make a Python script to extract pixel values within sky annulus.

Python Code 9: ao2021_s11_09.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
import astropy.modeling

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

# importing matplotlib module
import matplotlib.pyplot

# constructing parser object
desc = 'extraction of pixel values within sky annulus'
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)

# making plot
matplotlib.pyplot.xlabel ("X [pixel]")
matplotlib.pyplot.ylabel ("Y [pixel]")
matplotlib.pyplot.imshow (skyannulus_maskeddata, origin='lower', \
                          interpolation='nearest')
matplotlib.pyplot.colorbar ()
matplotlib.pyplot.savefig (file_output, dpi=225)
```

Run the script.

```
% chmod a+x ao2021_s11_09.py
% ./ao2021_s11_09.py -h
usage: ao2021_s11_09.py [-h] [-f FWHM] [-a APERTURE] [-s1 SKYANNULUS1]
                        [-s2 SKYANNULUS2] [-w WIDTH] [-x XCENTRE] [-y YCENTRE]
                        [-o OUTPUT]
                        file

extraction of pixel values within sky annulus

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_s11_09.py -f 5.03 -a 2.0 -s1 3.0 -s2 6.0 -w 35 -x 500.00 -y 499.99 \
? -o skyannulus_0.png synstars_0.fits
% ls -l skyannulus_0.png
-rw-r--r--  1 daisuke  taiwan  59377 Apr 22 23:55 skyannulus_0.png
```

Show the plot. (Fig. 8)

```
% feh -dF skyannulus_0.png
```

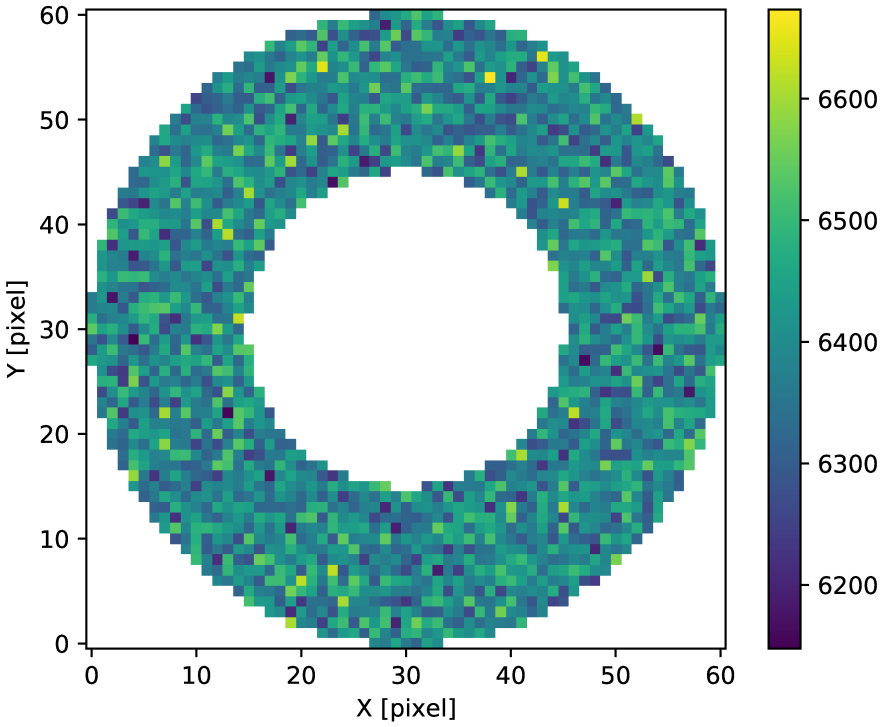

Figure 8: Extracted pixels within sky annulus.

5.2 Sky background level estimate using sigma-clipping algorithm

We can now calculate sky background level using sigma-clipping algorithm.

Python Code 10: ao2021_s11_10.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
import astropy.modeling
import astropy.stats

# importing photutils module
```

```
import photutils.centroids
import photutils.aperture

# 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 ('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_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 ()

# 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')

# printing result
print ("skybg_mean   = %f" % skybg_mean)
print ("skybg_median = %f" % skybg_median)
print ("skybg_stddev = %f" % skybg_stddev)
```

Execute the script, and show the result.

```
% chmod a+x ao2021_s11_10.py
% ./ao2021_s11_10.py -h
usage: ao2021_s11_10.py [-h] [-f FWHM] [-a APERTURE] [-s1 SKYANNULUS1]
                        [-s2 SKYANNULUS2] [-w WIDTH] [-x XCENTRE] [-y YCENTRE]
                        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

% ./ao2021_s11_10.py -f 5.03 -a 2.0 -s1 3.0 -s2 6.0 -w 35 -x 500.00 -y 499.99 \
? synstars_0.fits
skybg_mean   = 6396.147665
skybg_median = 6394.197813
skybg_stddev = 77.875421
```

The median value of sigma-clipped data is 6394 ADU.

5.3 Sky background subtraction using median of sigma-clipped data

Now, we are ready to carry out sky background subtraction using median of sigma-clipped pixel values within sky annulus.

Python Code 11: ao2021_s11_11.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
import astropy.modeling
import astropy.stats

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

# 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 ('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_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 ()

# 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) )
```

Execute the script, and show the result of aperture photometry.

```
% chmod a+x ao2021_s11_11.py
% ./ao2021_s11_11.py -h
usage: ao2021_s11_11.py [-h] [-f FWHM] [-a APERTURE] [-s1 SKYANNULUS1]
                        [-s2 SKYANNULUS2] [-w WIDTH] [-x XCENTRE] [-y YCENTRE]
```

```
                        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

% ./ao2021_s11_11.py -f 5.03 -a 2.0 -s1 3.0 -s2 6.0 -w 35 -x 500.00 -y 499.99 \
? synstars_0.fits
net flux = 153506.784646 +/- 1478.675470
```

5.4 Aperture photometry using smaller aperture radius

Use smaller radius for aperture, and do aperture photometry again.

Python Code 12: ao2021_s11_12.py

```python
#!/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)
```

Run the script using the aperture radius of 1.5 times of FWHM.

```
% chmod a+x ao2021_s11_12.py
% ./ao2021_s11_12.py -h
usage: ao2021_s11_12.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_s11_12.py -f 5.03 -a 1.5 -s1 3.0 -s2 6.0 -w 35 -x 500.00 -y 499.99 \
? -o aperture_2.png synstars_0.fits
net flux = 150492.912612 +/- 1137.559225
% ls -l aperture_2.png
```

```
-rw-r--r--  1 daisuke  taiwan  127812 Apr 23 00:38 aperture_2.png
```

The result of the aperture photometry is somewhat closer to the expected value compared to the one from larger aperture radius.

Show the location and size of the aperture. (Fig. 9)

```
% feh -dF aperture_2.png
```

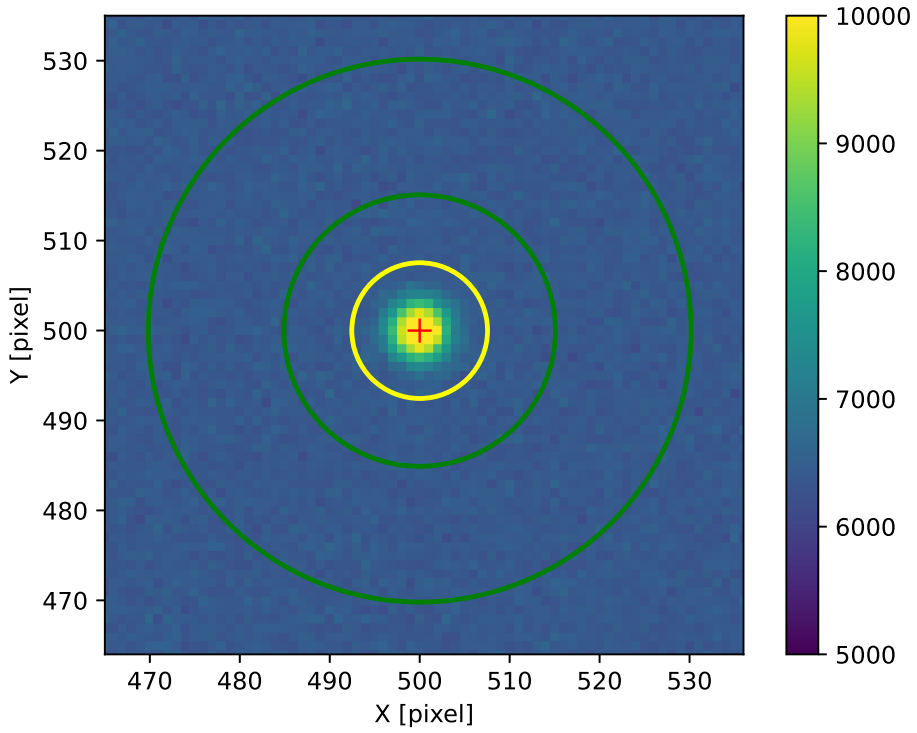

Figure 9: Location and size of aperture for star.

6 For your training

1. Read chapter 5 of "Handbook of CCD Astronomy" and learn about aperture photometry.
   - Handbook of CCD Astronomy (2nd Edition)
     - Steve B. Howell
     - Cambridge University Press
     - `https://doi.org/10.1017/CBO9780511807909`
2. Read the paper "Basic Photometry Techniques" and learn about aperture photometry.
   - "Basic Photometry Techniques"
     - Da Costa, G. S.
     - Astronomical CCD observing and reduction techniques, edited by Steve B. Howell.
     - 1992
     - `http://adsabs.harvard.edu/full/1992ASPC...23...90D`

7 Assignment

1. What is aperture photometry?
   (a) Describe the method of aperture photometry. What is the basic idea of aperture photometry? Why do we need to set sky annulus in addition to aperture for star?
   (b) What are pros and cons of aperture photometry?
   (c) What are commonly used photometry techniques other than aperture photometry? Under what circumstances we should use photometry techniques other than aperture photometry?
2. What are pros and cons of larger aperture radius?
3. What are pros and cons of smaller aperture radius?
4. For brighter sources, should we use larger aperture raduis, or smaller aperture radius? Describe your answer.
5. Typically, 1.5 times of FWHM or 2.0 times of FWHM is a good choice for aperture radius. Describe why.
6. For fainter sources, should we use larger aperture raduis, or smaller aperture radius? Describe your answer.
7. For 24 artificial stars other than the one you have measured on the image `synstars_0.fits`, make your own Python script to carry out aperture photometry using `photutils`. Show the source code of your Python script. Execute the script, and show the results. Examine your results.
8. Making one more synthetic image and doing aperture photometry.
   (a) Create a synthetic image using Python script `ao2021_s11_13.py`. The image should look like Fig. 10. You see fainter stars around the brighter star at (x, y) = (500, 500).
   (b) Make your own Python script to carry out aperture photometry of a star centred on (x, y) = (500, 500). Use simple mean of pixel values within sky annulus to estimate sky background level. Show the source code of your Python script. Execute the script, and show the result of sky background level estimate and aperture photometry.
   (c) Make your own Python script to carry out aperture photometry of a star centred on (x, y) = (500, 500). Use median of sigma clipped data of pixel values within sky annulus to estimate sky background level. Show the source code of your Python script. Execute the script, and show the result of sky background level estimate and aperture photometry.
   (d) Does sigma-clipping algorithm give a better estimate of sky background level? If so, describe why.
9. Assume 2-dimensional Gaussian function for the stellar PSF. When we set the aperture radius to be 1.5 times of FWHM, what is the fraction of stellar flux falling within the aperture? Make a Python script to calculate it. Show the source code of your Python script. Execute the script, and show the result.
10. Assume 2-dimensional Gaussian function for the stellar PSF. When we set the aperture radius to be 2.0 times of FWHM, what is the fraction of stellar flux falling within the aperture? Make a Python script to calculate it. Show the source code of your Python script. Execute the script, and show the result.

Python Code 13: ao2021_s11_13.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
import astropy.table
```

```
# importing photutils module
import photutils.datasets

# constructing parser object
desc = 'generating a synthetic image'
parser = argparse.ArgumentParser (description=desc)

# adding command-line arguments
parser.add_argument ('-b', '--background', type=float, default=2500.0, \
                     help='background level (default: 2500)')
parser.add_argument ('-s', '--sigma', type=float, default=50.0, \
                     help='noise level (default: 50)')
parser.add_argument ('-n', '--nstar', type=int, default=10, \
                     help='number of stars to generate (default: 10)')
parser.add_argument ('-f1', '--flux1', type=float, default=100000.0, \
                     help='total flux of bright star (default: 100000)')
parser.add_argument ('-f2', '--flux2', type=float, default=30000.0, \
                     help='total flux of faint star (default: 30000)')
parser.add_argument ('-p', '--psf', type=float, default=5.0, \
                     help='FWHM of stellar radial profile (default: 5)')
parser.add_argument ('-o', '--output', default='', \
                     help='output file name')

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

# input parameters
sky_background_level = args.background
noise_level          = args.sigma
nstar                = args.nstar
flux_total_1         = args.flux1
flux_total_2         = args.flux2
psf_fwhm             = args.psf
file_output          = args.output

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

# image size
image_size_x = 1024
image_size_y = 1024
image_size   = (image_size_x, image_size_y)

# location of bright star
x_centre = 500.0
y_centre = 500.0

# region to make stars
half_width = 30.0
x_min = x_centre - half_width
x_max = x_centre + half_width
y_min = y_centre - half_width
y_max = y_centre + half_width
```

```
list_x    = numpy.array ([x_centre])
list_y    = numpy.array ([y_centre])
list_flux = numpy.array ([flux_total_1])
list_psf  = numpy.array ([psf_fwhm])

while ( len (list_x) < nstar + 1 ):
    x = numpy.random.uniform (x_min, x_max)
    y = numpy.random.uniform (y_min, y_max)
    dist = numpy.sqrt ( (x - x_centre)**2 + (y - y_centre)**2 )
    if (dist < psf_fwhm * 3.0):
        continue
    list_x    = numpy.append (list_x, x)
    list_y    = numpy.append (list_y, y)
    list_flux = numpy.append (list_flux, flux_total_2)
    list_psf  = numpy.append (list_psf, psf_fwhm)

# generating sky background
image_background \
    = photutils.datasets.make_noise_image (image_size, \
                                           distribution='gaussian', \
                                           mean=sky_background_level, \
                                           stddev=noise_level)

# making a source table
fwhm_sigma = 2.0 * numpy.sqrt (2.0 * numpy.log (2.0) )
source_table = astropy.table.Table ()
source_table['x_0']   = list_x
source_table['y_0']   = list_y
source_table['flux']  = list_flux
source_table['sigma'] = list_psf / fwhm_sigma

# printing source table
print ("source_table:")
print (source_table)

# generating stars
image_star \
    = photutils.datasets.make_gaussian_prf_sources_image (image_size, \
                                                          source_table)

# making synthetic image by adding background and stars
image = image_background + image_star

# writing a FITS file
print ("Now, writing data into FITS file \"%s\"..." % file_output)
astropy.io.fits.writeto (file_output, image)
print ("Finished writing data!")
```

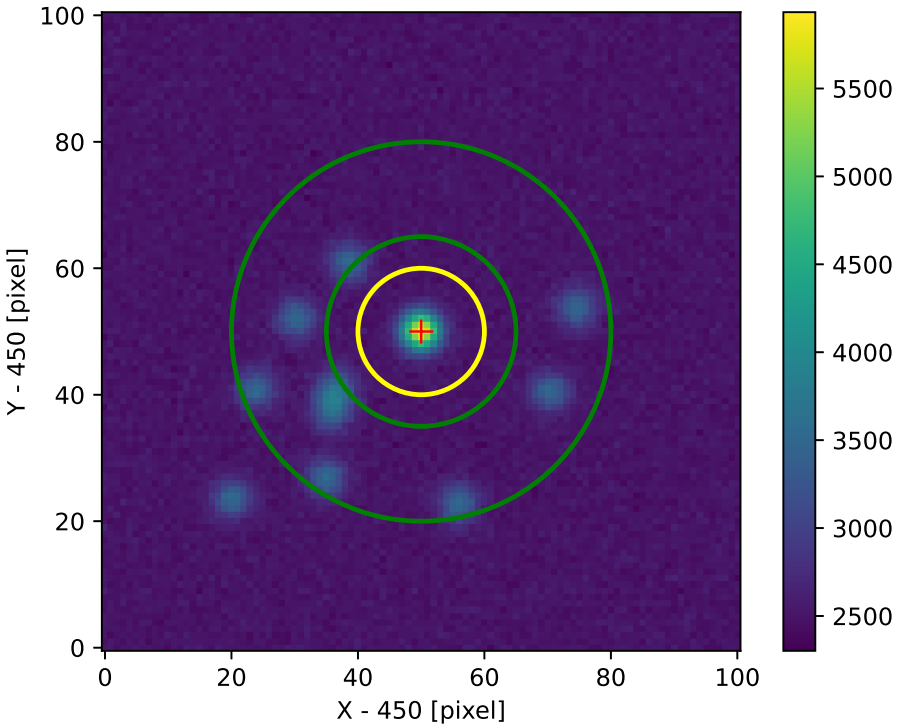

Figure 10: The synthetic image created by the script `ao2021_s11_13.py`.