GradStellar

Spectrum

How Far are the Stars?

GradStellar

Spectrum

Team and Roles

Mohammad Arshad	Team Leader
Akash Kumar	Team Member

GradStellar

About Project

In Astronomy, there exists numerous data on the stars and it needs to be well organized based on the requirement. We would like to organize the data based on the distance of the star.

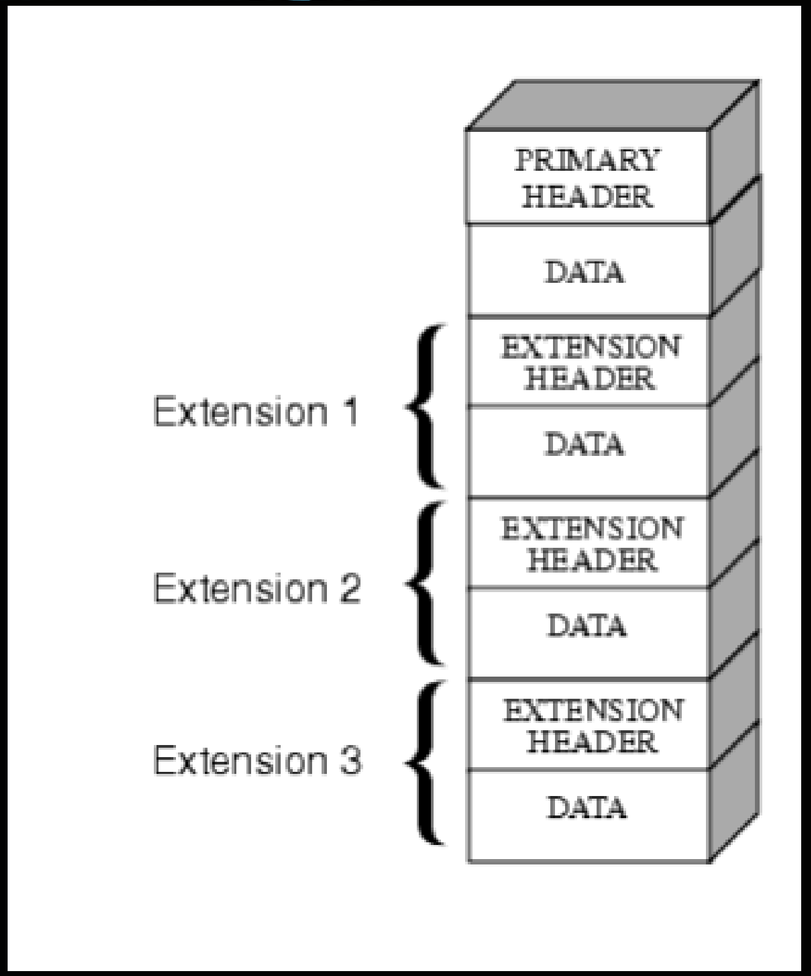
Flexible Image Transport System (FITS) is the most commonly used digital file format in astronomy. We are using the fits file of NGC5866 also called Messier 102 to identify the stars and measure the distance of the target stars through a calibration star.

We used different python libraries like **astroPy, sep, fitsio**.

GradStellar

Structure of FITS file

- A data file in FITS format consists of a series of Header Data Units (HDUs).
- Each of the HDUs contain 2 components ASCII text header and the binary data.
- Each extension can contain one of several different data types, including images, binary tables, ASCII text tables.



GradStellar

Spectrum

Primary and Image Header

```
Filename: frame-u-006122-1-0013.fits
No.    Name      Ver    Type      Cards   Dimensions   Format
  0  PRIMARY       1 PrimaryHDU      96   (2048, 1489)   float32
  1                1 ImageHDU         6   (2048,)   float32
  2                1 BinTableHDU     27   1R x 3C   [49152E, 2048E, 1489E]
  3                1 BinTableHDU     79   1R x 31C   [J, 3A, J, A, D, D, 2J, J, D, D, D, D, D, D, D, D, D, D,
D, D, D, D, D, D, D, D, D, D, D, E, E]
```

- RA   =   225.35840 / 1st row RA of telescope boresight (deg)
- DEC   =   56.733120 / 1st row Dec of telescope boresight (degrees)
- ORIGIN = 'SDSS   '
- TELESCOP= '2.5m   '
- RUN   =   6122 / Run number
- FRAME =   17 / Frame sequence number within the run

```
: XTENSION= 'IMAGE   '           /Image Extension created by MWRFITS v1.11
  BITPIX  =                   -32 /
  NAXIS   =                     1 /
  NAXIS1  =                  2048 /
  PCOUNT  =                     0 /
  GCOUNT  =                     1 /
```

GradStellar

Spectrum

Astropy

- Astropy is a collection of software packages written in python programming language and designed for use in Astronomy.
- FITS files can only be viewed online, only some fits files can be downloaded. Through astropy module we use the inbuilt function called "download_file" to return the fits file.

```
from astropy.utils.data import download_file
from astropy.io import fits
def get_data(filename):
    file = filename
    image_file = download_file(file)
    data = fits.getdata(image_file)
    data = data.byteswap().newbyteorder()
    return data
```

GradStellar

Implementation

```
data = fitsio.read('frame-u-006122-1-0013.fits')
```

```
data
```

```
array([[-0.00439453, -0.02294922, -0.03222656, ..., -0.01235962,
        -0.02151489,  0.05181885],
       [ 0.00489044,  0.00488281, -0.00439453, ...,  0.01515198,
        -0.01235962,  0.05181885],
       [ 0.02346802,  0.03271484, -0.04150391, ...,  0.04266357,
        -0.03070068, -0.03070068],
       ...,
       [ 0.0055542 ,  0.01483154,  0.02410889, ..., -0.01173401,
        -0.05755615, -0.03005981],
       [ 0.0241394 , -0.01300049, -0.04083252, ...,  0.07995605,
         0.00660706, -0.04840088],
       [-0.02230835, -0.05010986, -0.05938721, ...,  0.01577759,
         0.0524292 ,  0.01577759]], dtype=float32)
```

- Using fitsio module we read the fits file as data.

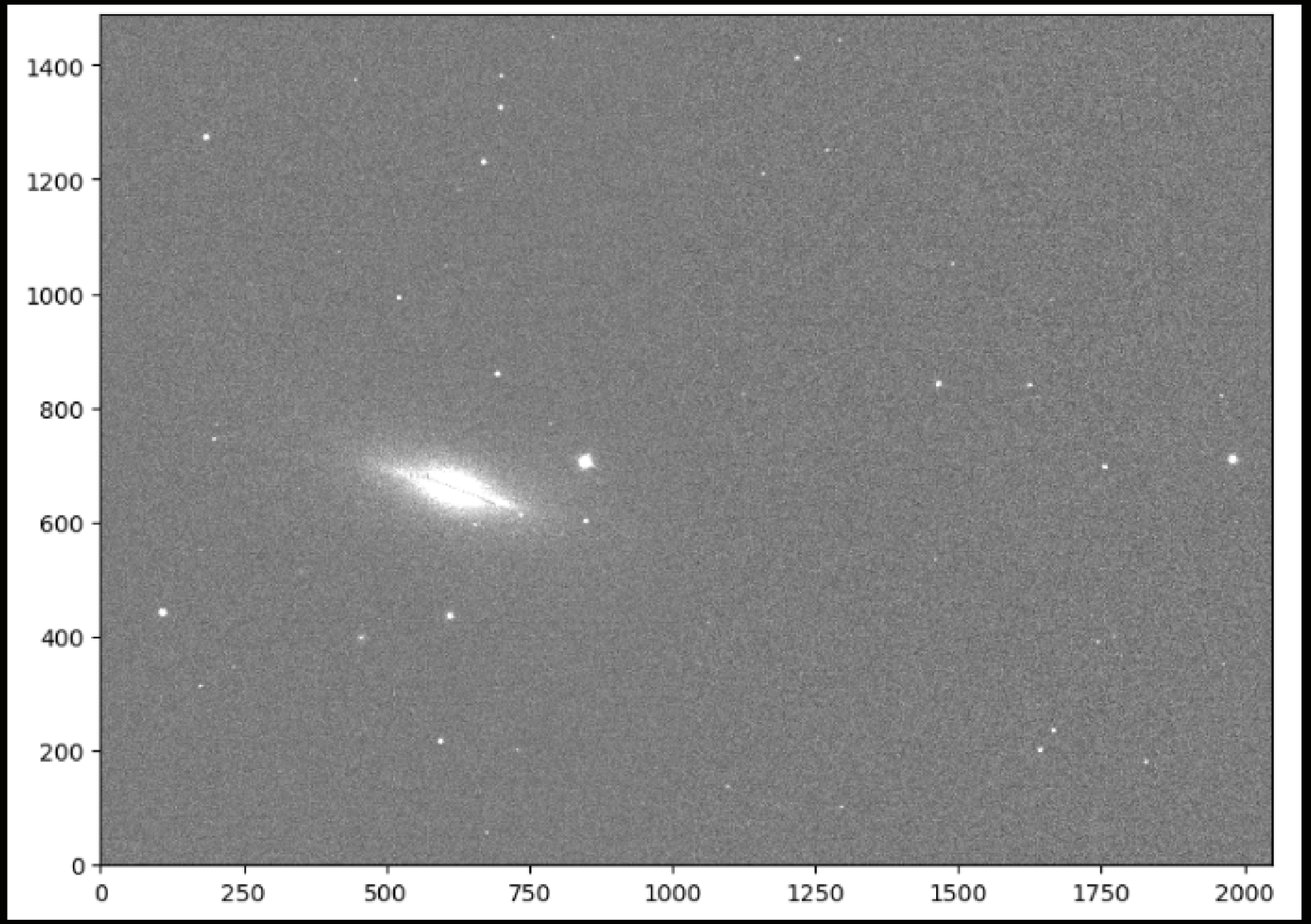
- Through sep.Background() we return the background data in the form of 2D array.

```
def subtract_background(data):
    bkg = sep.Background(data)
    bkg_image = bkg.back()
    bkg_rms = bkg.rms()
    data_sub = data - bkg
    return data_sub, bkg
```

GradStellar

```
data_sub, bkg = subtract_background(data)
bkg_image = bkg.back()
```

```
m1, s1 = np.mean(data), np.std(data)
plt.imshow(data, interpolation='nearest', cmap='gray',vmin=m1-s1, vmax=m1+s1, origin='lower')
```

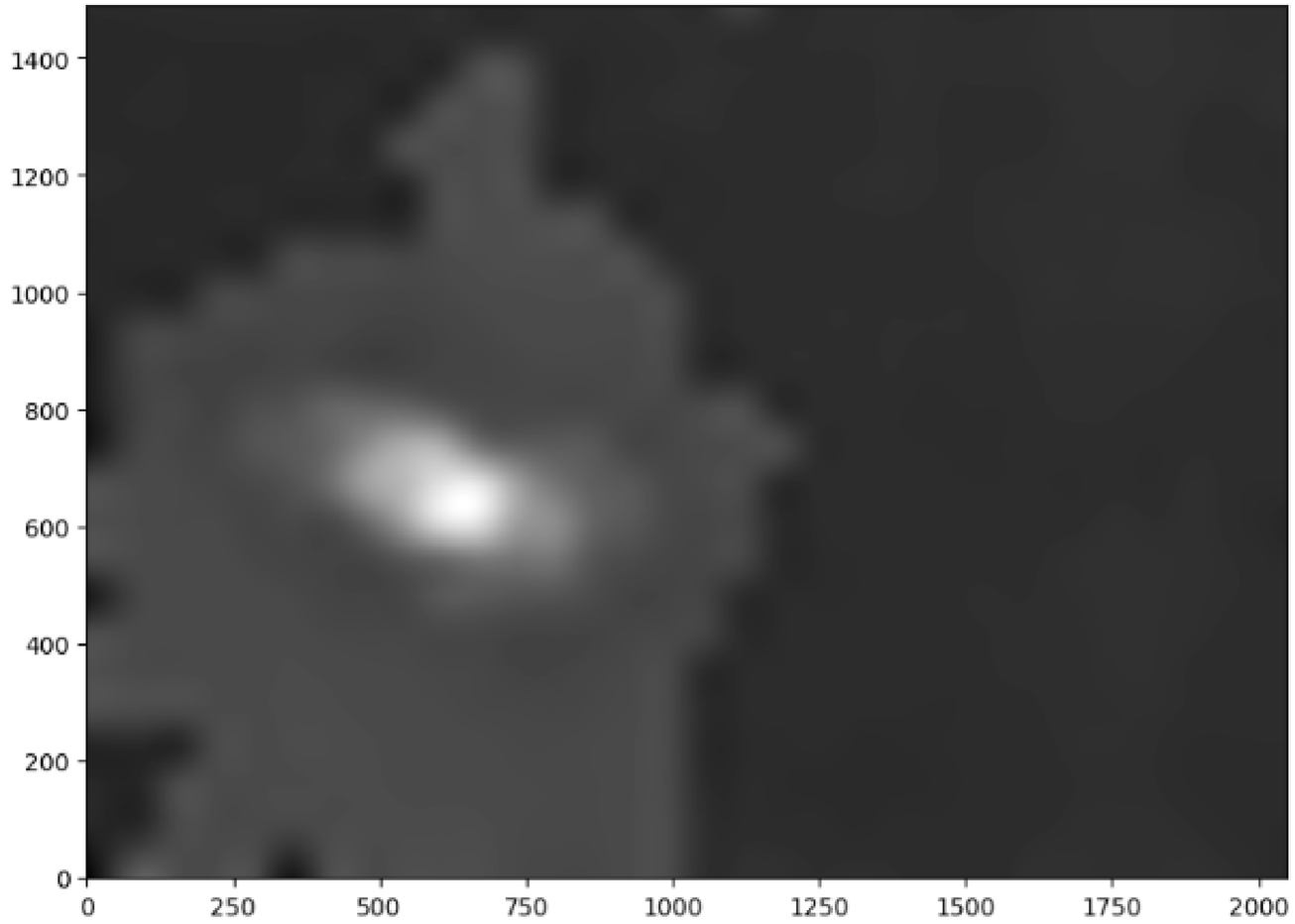


GradStellar

Background Noise

```
plt.imshow(bkg_image, interpolation='nearest', cmap='gray', origin='lower')
```

```
<matplotlib.image.AxesImage at 0x7f1d2b875310>
```

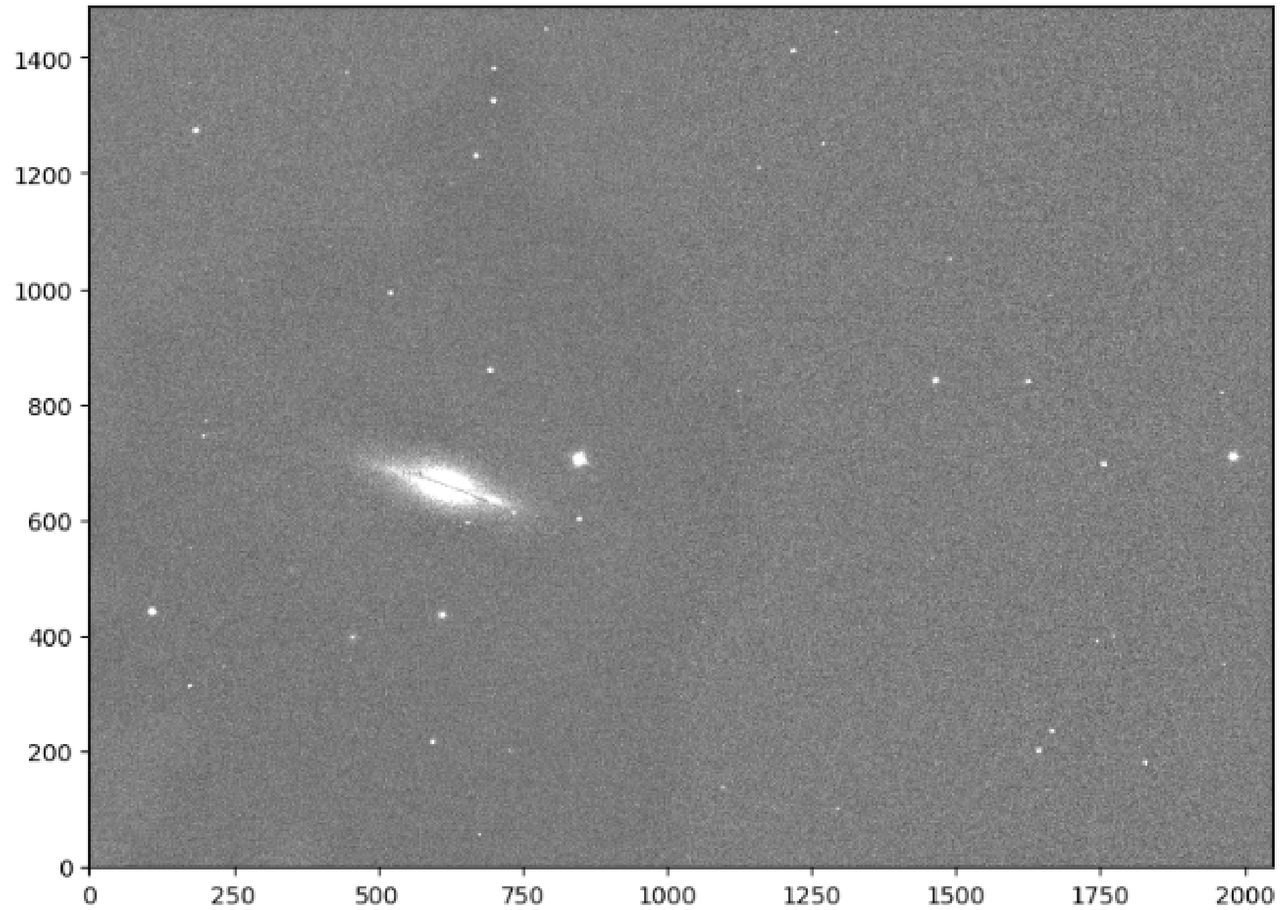


GradStellar

Noise Free

```
m2, s2 = np.mean(data_sub), np.std(data_sub)
plt.imshow(data_sub, interpolation='nearest', cmap='gray',vmin=m2-s2, vmax=m2+s2, origin='lower')
```

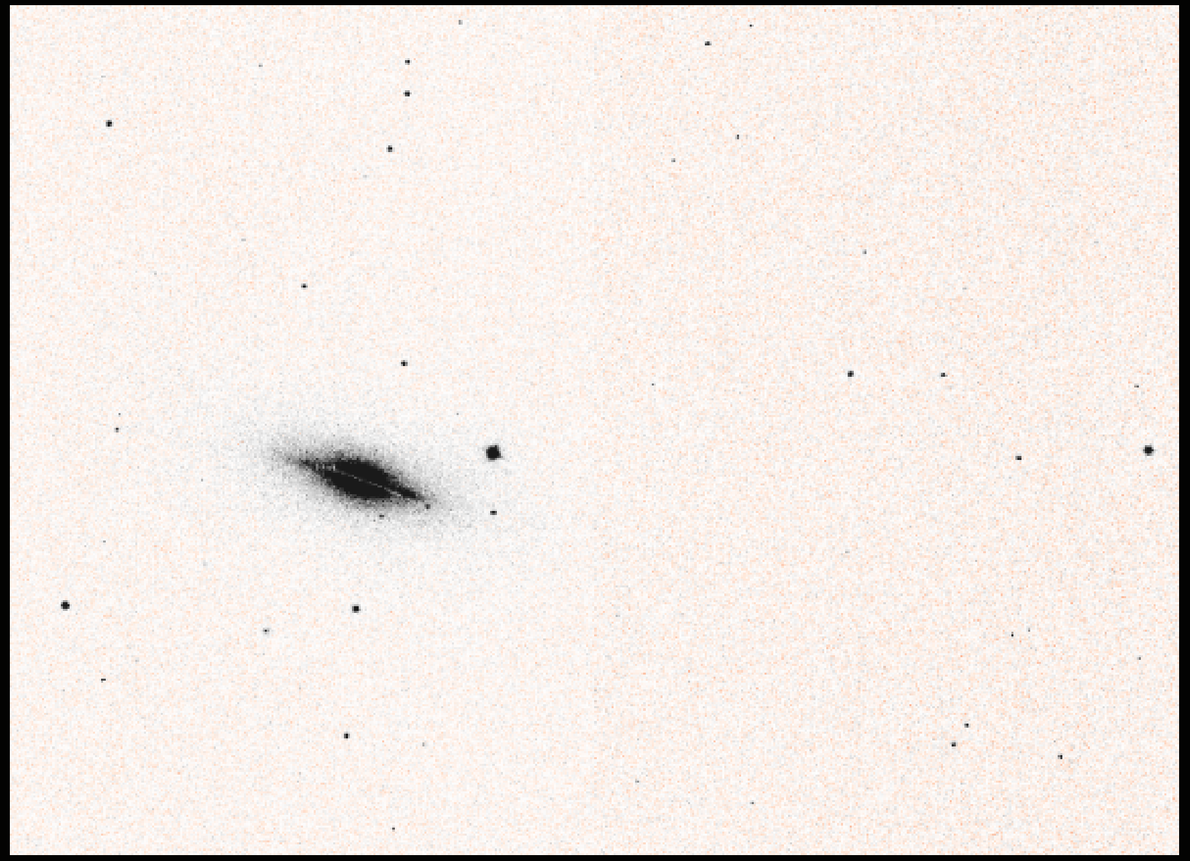
```
<matplotlib.image.AxesImage at 0x7f1d2b73bd60>
```



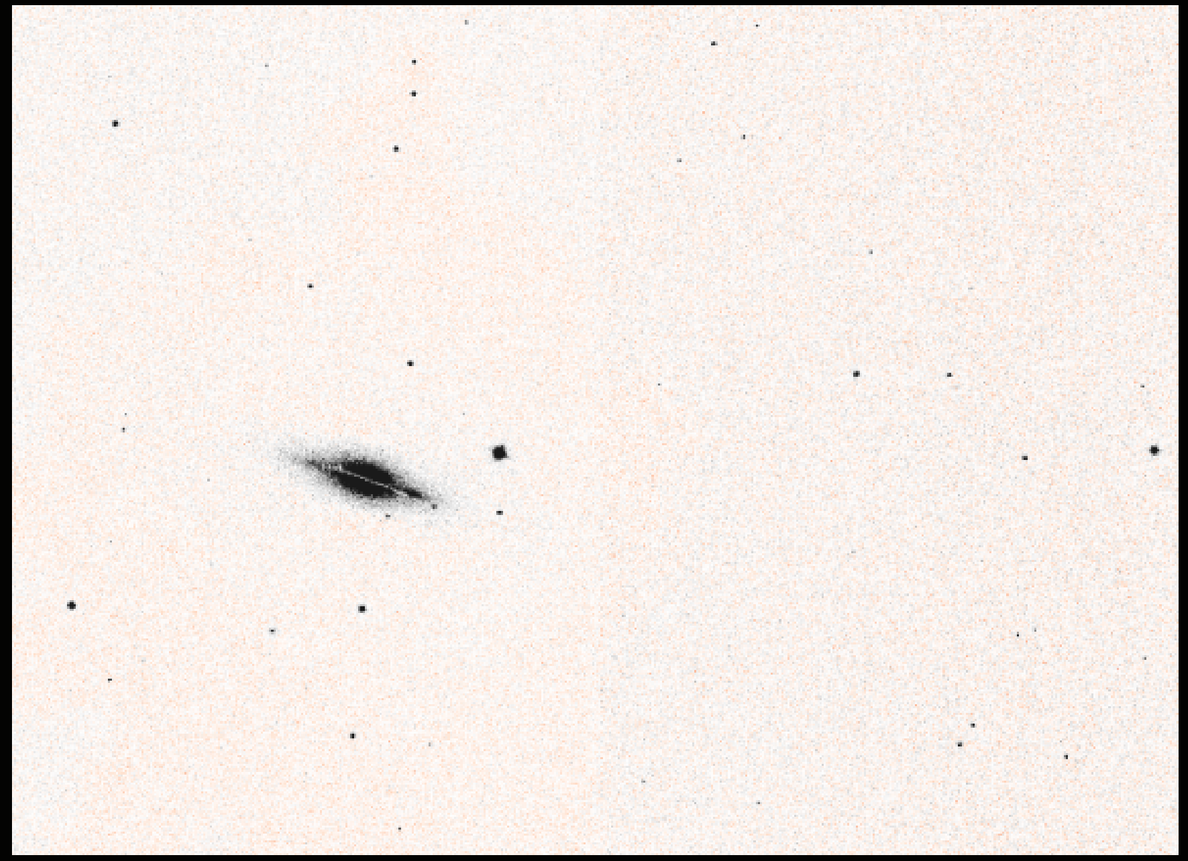
GradStellar

Spectrum

Comparison



Original Image



Noise-free Image

GradStellar

Objects

```
objects = sep.extract(data_sub, 1.5, err=bkg.globalrms)

#sep.extract returns a numpy array called Objects which has 30 data types
```

- Using sep.extract() we obtained the sources having a flux 1.5 times the global rms.
- sep.extract() returns an object containing 30 data types.
- We obtain say the data of flux through the syntax **objects['flux'].**

```
objects.dtype.names
```

```
('thresh',
 'npix',
 'tnpix',
 'xmin',
 'xmax',
 'ymin',
 'ymax',
 'x',
 'y',
 'x2',
 'y2',
 'xy',
 'errx2',
 'erry2',
 'errxy',
 'a',
 'b',
 'theta',
 'cxx',
 'cyy',
 'cxy',
 'cflux',
 'flux',
 'cpeak',
 'peak',
 'xcpeak',
 'ycpeak',
 'xpeak',
 'ypeak',
 'flag')
```

```
print((objects['flux']))
print(len(objects['flux']))
```

```
[3.40845895e+00 8.81644344e+00 1.22643244e+00 4.94853675e-01
 6.51145041e-01 9.40634537e+00 7.68588960e-01 5.40374160e-01
 1.08799160e+00 9.16043937e-01 1.00493050e+01 1.11934817e+00
 4.45627022e+01 7.50412405e-01 6.33776760e+00 8.33466034e+01
 7.90010929e-01 1.68897537e+02 3.21058960e+01 6.26049995e-01
 5.80442548e-01 2.29075956e+00 1.97777605e+00 9.79131520e-01
 1.21432567e+00 1.35970783e+01 4.48713869e-01 4.91127700e-01
 4.41235876e+00 3.46785069e+00 6.42901611e+00 7.86722124e-01
 1.67482567e+01 6.29827595e+00 2.43138981e+01 6.50815308e-01
 7.51325071e-01 4.53188717e-01 4.05081320e+00 5.72595337e+02
 6.00469112e-01 6.70353088e+02 4.49375004e-01 5.33509076e-01
 7.74651647e-01 1.83728755e+00 3.21353054e+00 6.30414009e-01
 7.12426376e+00 2.92883539e+00 5.79953432e-01 1.69791794e+00
 2.86063862e+01 6.39479280e-01 7.17948608e+01 3.83833337e+00
 6.45413756e-01 4.34177101e-01 3.22267222e+00 3.82132679e-01
 5.10997772e-01 4.19614017e-01 8.38849411e+01 2.87356901e+00
 1.18116345e+03 3.42995941e+02 2.51161841e+03 3.19377588e+03
 4.16897217e+03 1.10189829e+01 6.45246089e-01 3.07642698e+00
 5.37820673e+00 5.48750281e-01 2.56657434e+00 2.33340573e+00
 1.68509254e+01 3.75788784e+00 7.10809469e-01 4.08770790e+01
 2.22297058e+02 6.41590953e-01 1.73497437e+02 4.13760453e-01
 4.05565977e-01 4.57105160e-01 4.38358498e+00 6.61183395e+01
 1.14834189e+00 3.42574477e+00 2.60364795e+00 4.63405102e-01
 2.79862976e+00 6.41609967e-01 9.12434769e+00 3.92296886e+00
 2.83022952e+00 1.03933346e+00 5.98159671e-01 5.76588213e-01
 2.04147840e+00 1.54213619e+00 5.07944918e+00 7.07943916e-01
 1.21678753e+01 7.13409245e-01 1.95280701e+02 7.88491786e-01
 4.80116844e-01 1.25060511e+01 3.45589081e+02 8.97173166e-01
 8.07332218e-01 3.83919358e-01 1.06906068e+00 1.20050133e+02
 3.33763075e+00 4.78326917e-01 9.80841458e-01 1.89803612e+00
 9.97994900e+00 6.54314339e-01 3.95745926e+01 4.97164279e-01
 5.72679043e-01 5.20610886e+01 8.48204315e-01 1.09016390e+01
 1.10069752e+00 1.86868973e+01 3.95394832e-01 1.14763844e+00]
132
```

GradStellar

Spectrum

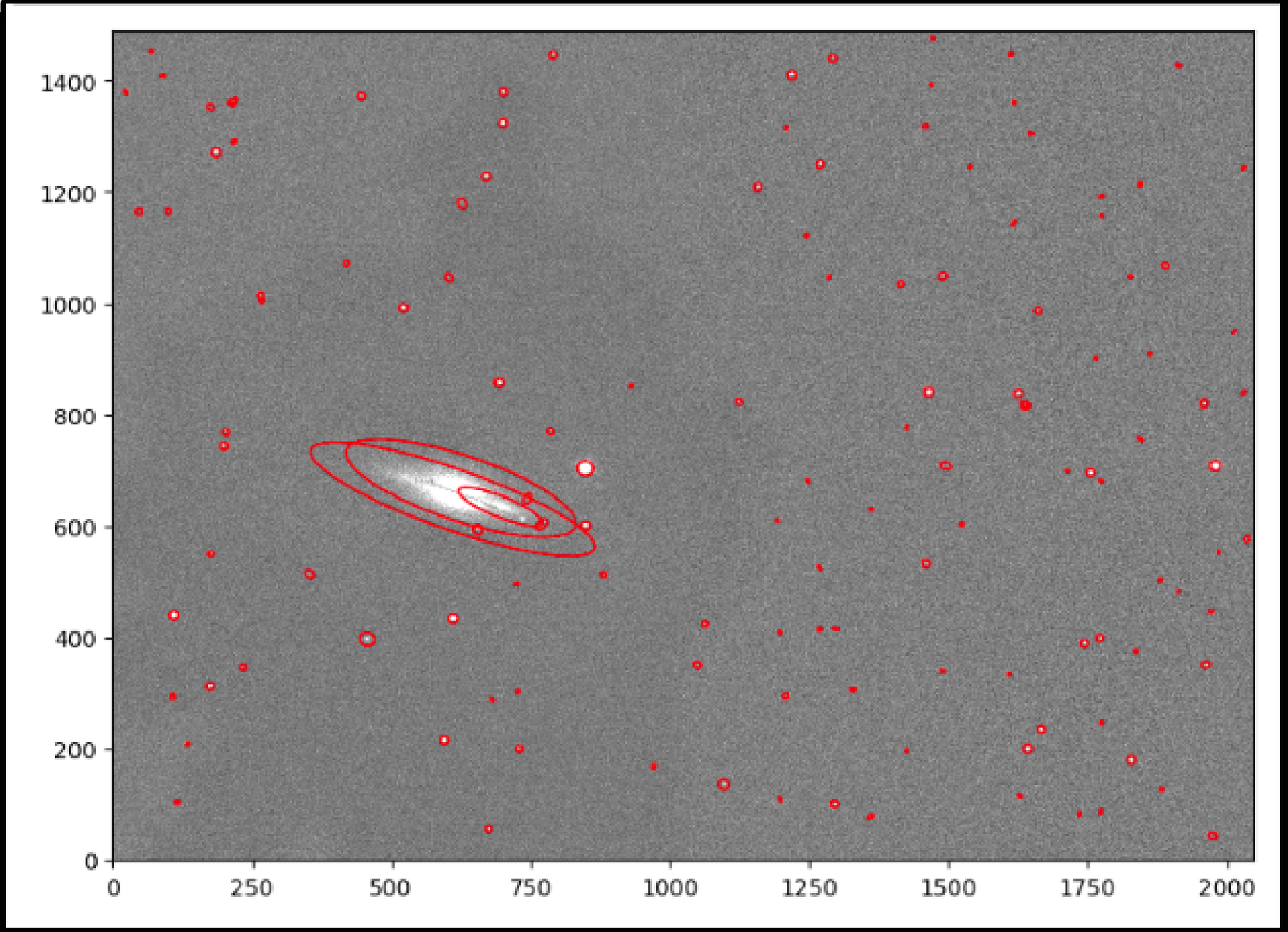
Source Detection

```
fig, ax = plt.subplots()
m, s = np.mean(data_sub), np.std(data_sub)
im = ax.imshow(data_sub, interpolation='nearest', cmap='gray', vmin=m-s, vmax=m+s, origin='lower')

for i in range(len(objects)):
    e = Ellipse(xy=(objects['x'][i], objects['y'][i]), width=9*objects['a'][i], height=9*objects['b'][i],
                angle = objects['theta'][i]*180.0/np.pi)
    e.set_facecolor('none')
    e.set_edgecolor('red')
    ax.add_artist(e)
```

Calibration Star

- Among the sources detected we choose a particular star as calibration star.
- The distance of this calibration star from the earth is known.
- We chose target star and find the distance from the calibration star.



GradStellar

Target

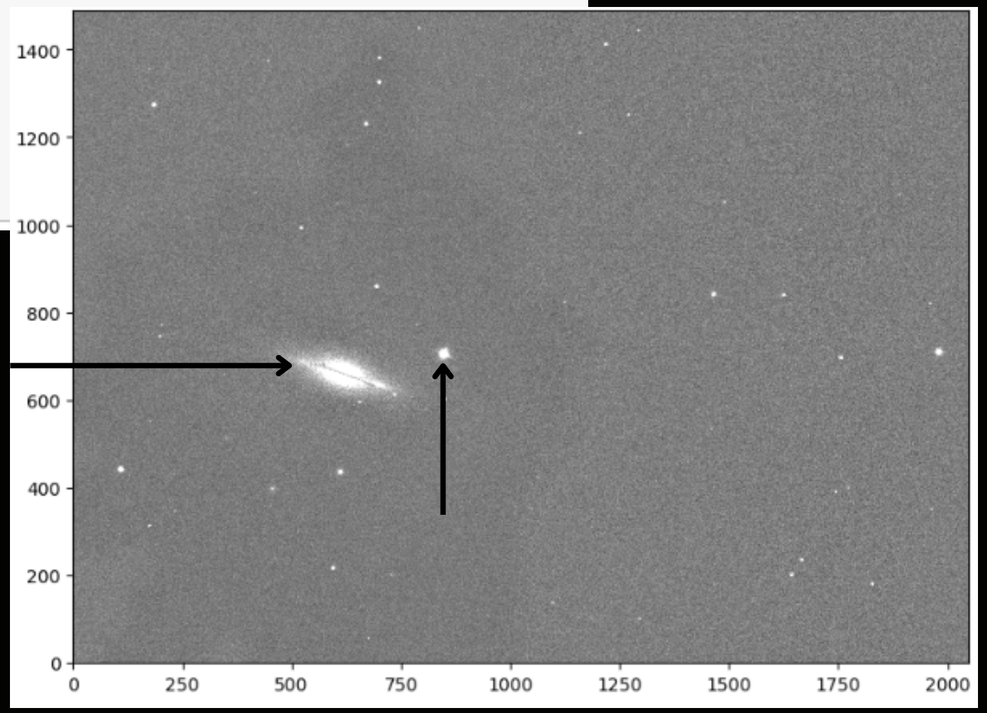
```
def target_max(xmin, xmax, ymin, ymax):
    A=[]
    x_coordinate=[]
    y_coordinate=[]
    for i in range(len(objects['x'])):
        if (objects['x'][i]< xmax and objects['x'][i]>xmin and objects['y'][i]>ymin and objects['y'][i]<ymax):
            A.append(objects['flux'][i])
            x_coordinate.append(objects['x'][i])
            y_coordinate.append(objects['y'][i])
    for i in range (len(A)):
        if A[i]== max(A):
            index = i
    maximum = max(A)
    return maximum, x_coordinate[index], y_coordinate[index]
```

```
target_max(500,750,600,800)
```

```
(3193.77587890625, 610.2371526110873, 647.9925536597518)
```

```
target_max(750,1000,600,800)
```

```
(4168.97216796875, 848.5173672782347, 702.8416338647834)
```



GradStellar

Distance Module

```
def distance_modulus(m,M):
    return 10**( (m-M)/5 +1)
def app_mag(flux):
    return -2.5*math.log10(flux/(25.11*10**8))
```

```
def target_distance(target_flux):
    calib_flux = 0.7685889601707458
    calib_flux_app = app_mag(calib_flux)
    f_cal = calib_flux_app / calib_flux
    m_target = app_mag(target_flux)
    M_target = m_target/f_cal
    return (10**( (m_target-M_target)/5 +1))
```

The parsec (symbol: pc) is a unit of length used to measure the large distances to astronomical objects outside the Solar System, approximately equal to 3.26 light-years

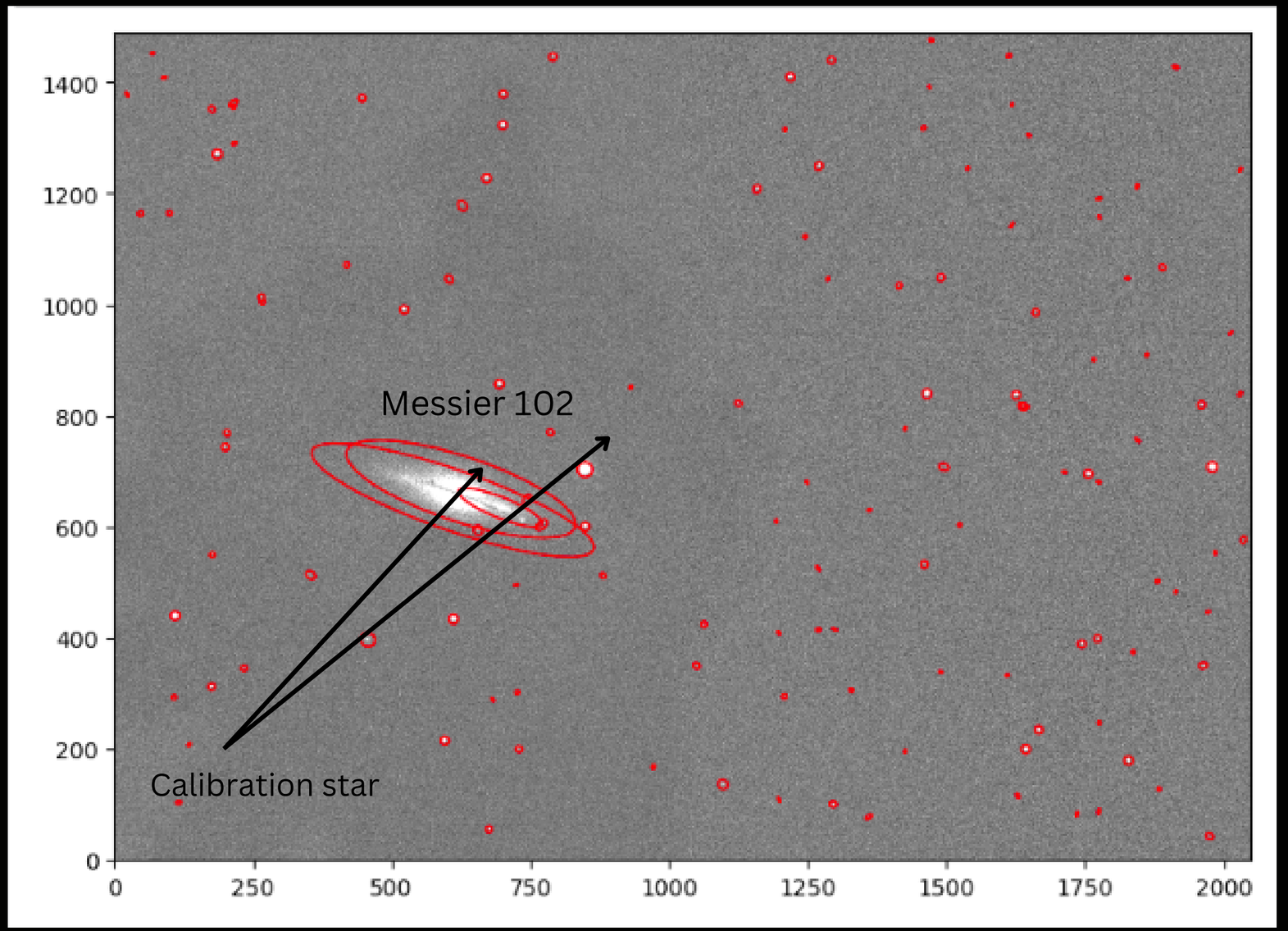
```
print(target_distance(4168.97216796875))
```

```
6259.300242140447
```

```
print(target_distance(3193.77587890625))
```

```
7120.628542307405
```

Distance unit is pc.


Messier 102
Calibration star
0
200
400
600
800
1000
1200
1400
0
250
500
750
1000
1250
1500
1750
2000

GradStellar

Company name