# H-R diagram of Messier 55

## I. Introduction

M55 (NGC 6809) is a popular subject for H-R diagrams as it's loose concentration and high galactic latitude (minimizing the interstellar reddening by dust which is densely packed in the galactic thin disk) makes it well suited for it. As such there have been multiple studies in the past on M55's H-R diagram by Alcaino et al. (1992), S-W Lee (1977), Zaggia et al. (1997) and more. We will make our own H-R diagram of M55 using data from Las Cumbres Observatory's (LCO) SBIG 6303 0.4m class telescope, process it with the astropy and photoutils python library and compare it to previously works.

## Table of contents

# II. Historical background and importance

## A. H-R diagram

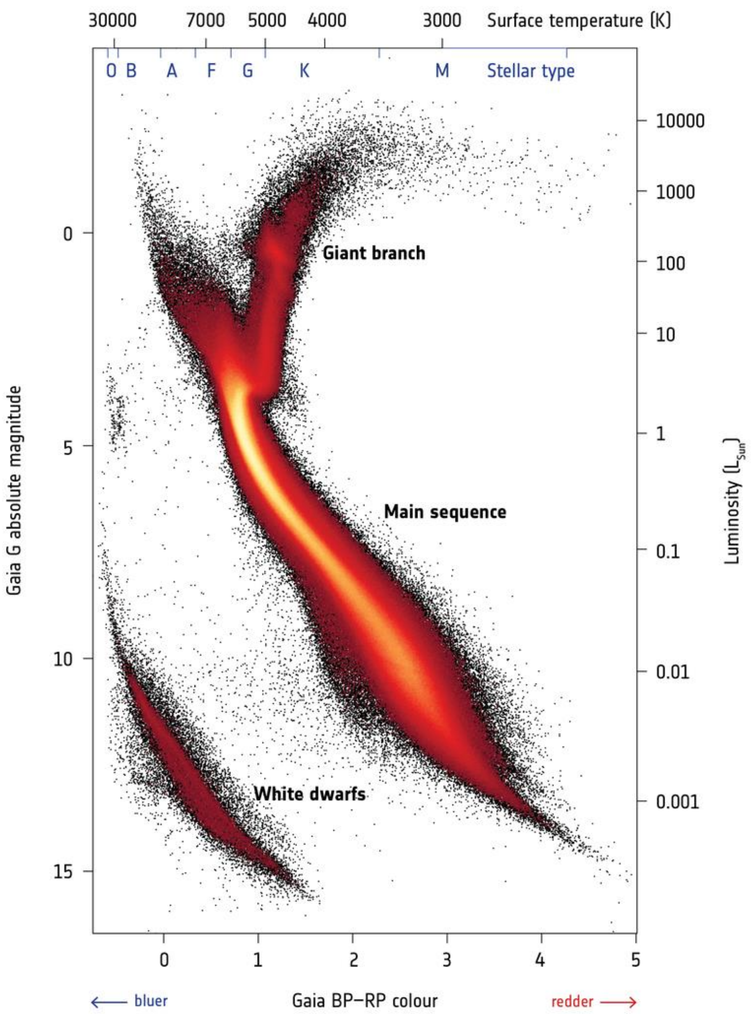



*Figure 1: Gaia's H-R diagram of the Milky Way, Credit: ESA/Gaia/DPAC, CC BY-SA 3.0 IGO*

A Hertzsprung-Russel diagram is a scatter plot showing the temperature/ luminosity against the stellar classification/ temperature/ color index for a stellar population. It was co-created independently in 1911 and 1913 by Danish astronomer Ejnar Hertzsprung and American astronomer Henry Norris Russel.

It is a useful tool to show the age and stage of evolution of a stellar population as well as it's composition. Most of the stars will be situated on the main sequence, a diagonal line going from top left (hotter, more luminous) to bottom right (colder, less luminous). At the end of their lives stars often "branch out" into the giant and supergiant branches which is located near the top left. The bottom right part of the diagram is reserved for white dwarfs, remnants of stars with mass typically $M < 8\ M_{\odot}$.

## B. Globular clusters

Globular clusters are masses of stars held together by gravity. A group of stars is usually considered a cluster from 10 stars onwards. These stellar objects are almost entirely free of other matter such as gas or dust and are found mostly in the outer parts, the halo of galaxies. They are of high astronomical interest because the stars that make these clusters up are some of the oldest known stars but also because their location in galaxies may clue us as to how these galaxies were formed.

Before the advent of telescopes, many globular clusters were thought to be stars or nebulae/galaxies. In 1665 that German astronomer Abraham Ihle discovered the first known globular cluster, M22, an object in the constellation Sagittarius with an apparent magnitude of 5.1. It was only in 1764 that individual stars in a cluster were resolved by Charles Messier when he observed M4. The term *globular cluster* first appeared in William Herschel's *Catalogue of a Second Thousand New Nebulae and Clusters of Stars* published in 1789. In 1914, Harlow Shapely used RR Lyrae Variable stars, at the time assumed to be Cepheid variable stars to estimate the distance to some of these clusters. In 1918, he also used this globular cluster distribution to estimate the position of our solar system in the Milky Way. Since then, the number of globular clusters that have been discovered in the Milky Way has only grown, up to 157 in 2010, with some speculated to be hidden away by the galactic center. It was also found that the number of globular clusters roughly correlates with the mass of the galaxy, with the Andromeda galaxy containing about 400 compared to the tens of thousands in the giant elliptical galaxy M87.

Globular clusters are on average made up of some of the oldest matter in the Universe. This fact can be seen in the low to very low metallicity, the proportion of heavy elements, of their stars. These globular clusters are also packed much more densely than the stellar average in the Milky Way. This leads to a higher proportion of "exotic" objects than normal such as binary systems composed of stars, white dwarfs, neutron stars and black holes, blue stragglers (exceptionally hot and luminous stars outside of the main sequence), extremely fast rotating pulsars, and low mass x-ray binaries (binary star systems with a star and either a black hole or a neutron star that emitting most of their light in the x-ray spectrum).

Globular clusters can be separated into classes using the Shapley–Sawyer Concentration Class (Shapely, Sawyer, 1927) with clusters having a high concentration towards the center being closer to I and those with a low central concentration being closer to XII.

*Figure 2: Palomar 12, a class XII star cluster, Credit: ESA/Hubble, NASA*

*Figure 3: Messier 75 a class I globular cluster, Credits: NASA, ESA, STScI, and G. Piotto (Università degli Studi di Padova) and E. Noyola (Max Planck Institut für extraterrestrische Physik)*

## C. Messier 55

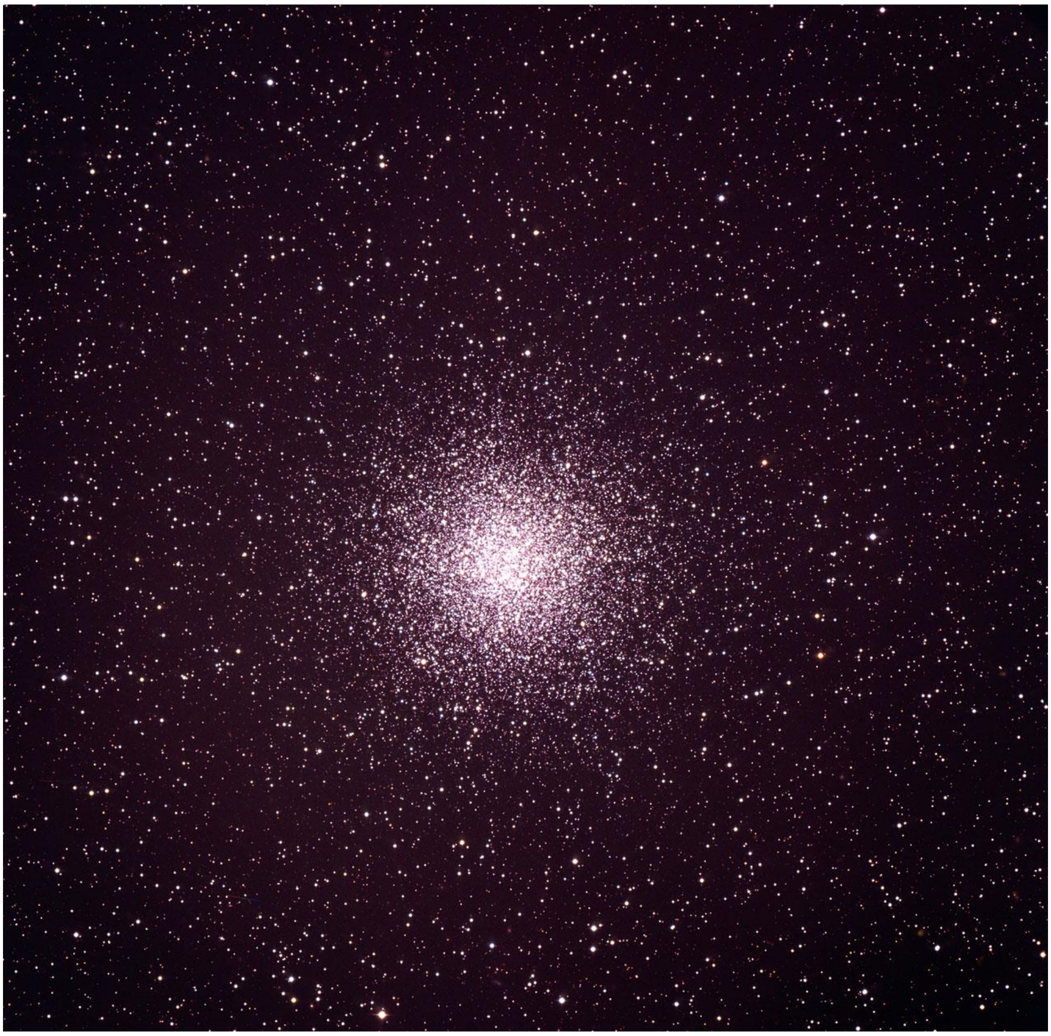


*Figure 4: M55 observed by ESO's 3.6m telescope in the optical band, Credit: ESO*

Messier 55 (NGC 6809 also known as the "Summer Rose Star") is a globular cluster located in the Sagittarius constellation. It is located 5.31kpc from Earth, has a mass of around 269.000 $M_\odot$ and occupies 19x19 arc minutes in the sky (data from Baumgart, Hilker, (2018)). The right ascension of this object is 19h 39m 59.71s and its declination is -30°57'53.1". It has a metallicity of:

$$\left[\frac{F_e}{H}\right] = \ log_{10}\left(\frac{N_{F_e}}{N_H}\right)_{M55} - log_{10}\left(\frac{N_{F_e}}{N_H}\right)_{Sun} \ = \ -1.94$$

With $N_{F_e}$ and $N_H$ the number of iron and hydrogen atoms per $m^3$. This makes it one of the objects in the Milky Way with the least metallicity and thus one of the oldest.

The central radius of M55 is 2.93pc, its half-mass radius is 6.92pc and its half-light radius is 4.70pc. It is estimated to have around 100.000 stars with very few variable stars.

This globular cluster was first discovered in 1752 by French astronomer Nicolas Louis de Lacaille who noted that "It resembles an obscure nucleus of a big comet. It was added to the Messier catalog a few decades later in 1778. The first to resolve the individual stars in this cluster is William Herschel in 1783.

### D. Importance of project

H-R diagrams are an important part of understanding stellar evolution as they outline how stars age depending on their mass/luminosity as well as how they form. For M55's H-R diagram and those of globular clusters in general, they are an important tool in understanding the birth and evolution of galaxies. The population of stars in these can give us insight into how globular clusters are created as well as help form theories as to past galactic encounters and collisions of the Milky Way.

## III. Image processing

### A. Observation request and signal to noise ratio

We used LCO's SBIG STL-6303 0.4m to observe M55 to take 100s exposure images in the Bessel B and Bessel V bands.

*Tableau 1: Observation request submitted to LCO*

| Target Name | RA | Dec | Filter | # Exposures | Integration Time (s) |
|---|---|---|---|---|---|
| M55 | 19:39:59.71 | -30:57:53.1 | Bessell V | 1 | 100 |
| M55 | 19:39:59.71 | -30:57:53.1 | Bessell B | 1 | 100 |

We found on SIMBAD that most of the stars in this cluster were between magnitude 10 and 20. Let's calculate the signal to noise ratio for stars in this interval. Using PHYS134L lecture notes, we can find an expression for the signal to noise ratio:

$$\frac{S}{N} = \frac{FA_\varepsilon\sqrt{\tau}}{\left[\frac{N_R^2}{\tau} + FA_\varepsilon + i_{DC} + F_\beta A_\varepsilon \Omega\right]^{1/2}}$$

Where F is the flux, $F_\beta$ the flux of the sky, $A_\varepsilon$ the telescope effective area, $\tau$ the integration time, $N_R$the readout noise and $\Omega$ the pixel size.

Since we have magnitudes and not flux, we need an expression for the conversion:

$$\frac{b_1}{b_2} = 10^{0.4\,(m_2-m_1)}$$

With $b_1$ and $b_2$ being the flux of two stars and $m_1$ and $m_2$ their magnitudes. We can use Vega as our reference star to find the flux of our desired object for a given magnitude. For Vega's Magnitude and flux in the UBVRI bands, we used data from Colina, Rohlin & Casteli (1996)'s work.

We can now plot the S/N ratio over magnitude and the S/N ratio over time:

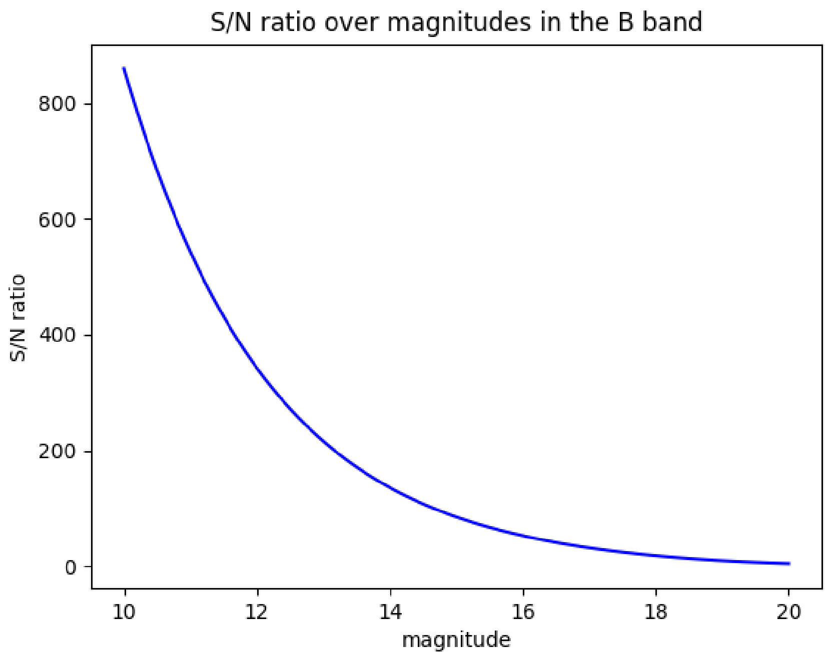



*Figure 5: S/N ratio over magnitudes in the B band*

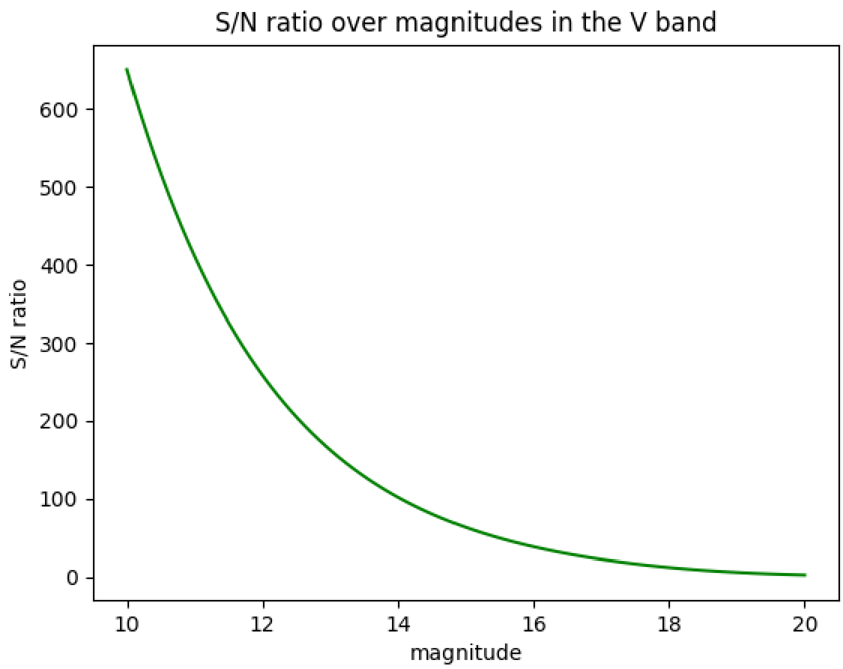



*Figure 6: S/N ratio over magnitude in the V band*

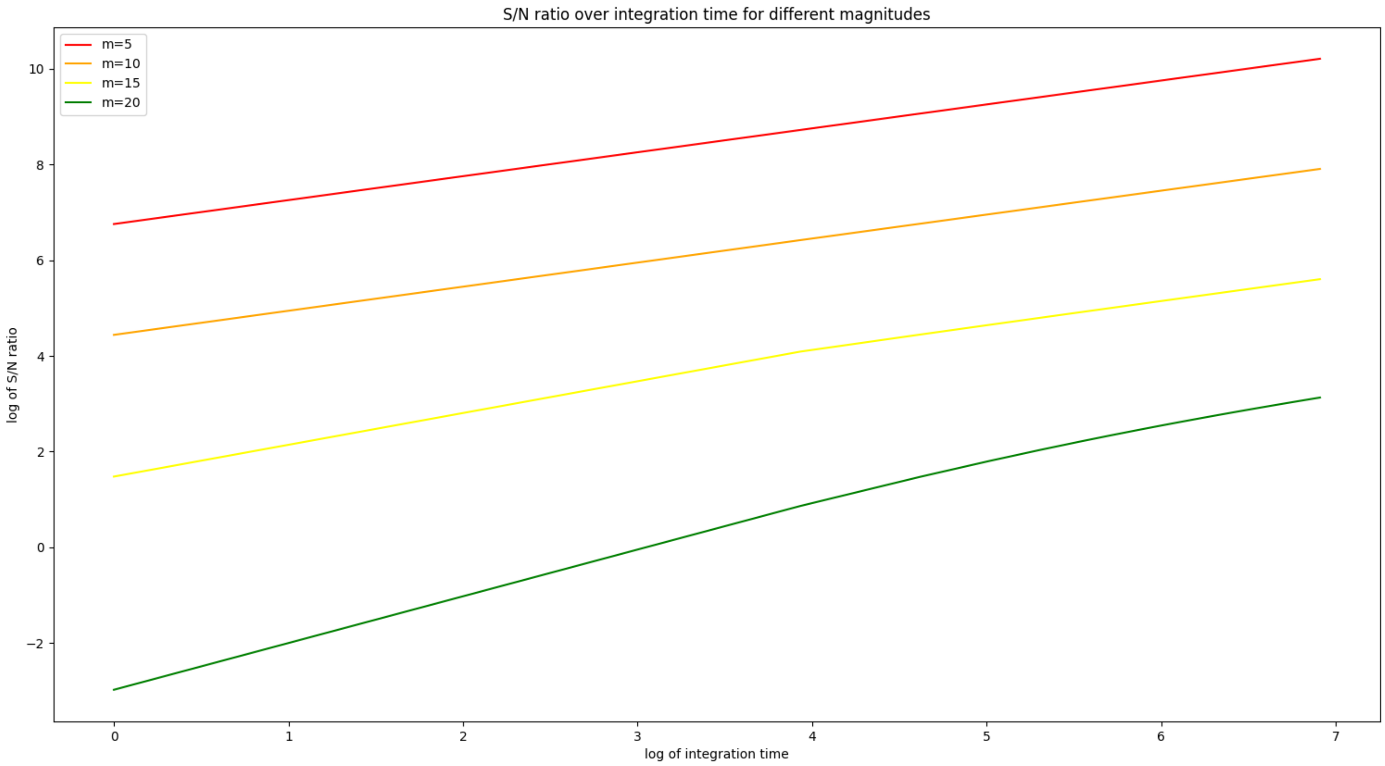


*Figure 7: SNR over integration time for different magnitudes*

As observed, we seem to have a decent S/N ratio all the way up to m = 19 where SNR is around 9 in the B band and around 6 in the V band.

## B. Visualizing the images

Opening the images in Astroart 8 gives us:

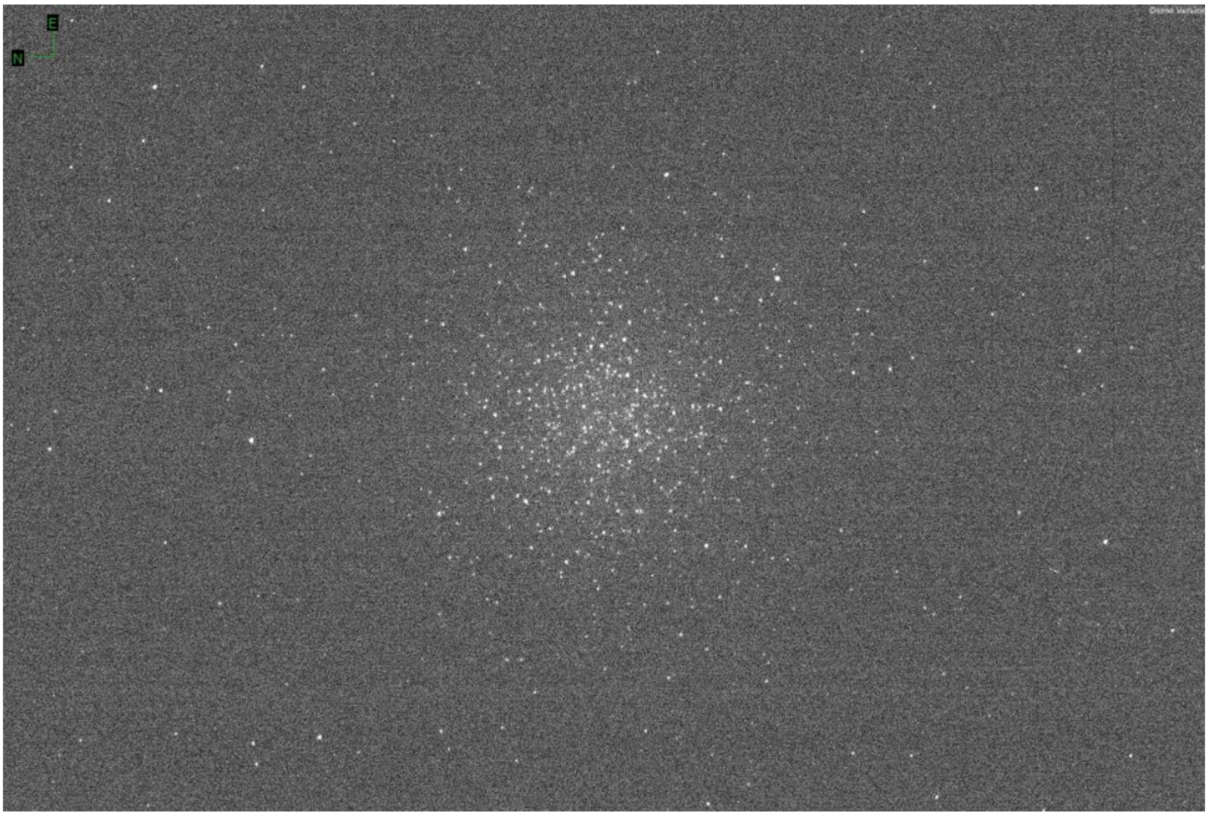



*Figure 8: images of M55 through the Bessel B filter opened through Astroart 8*


*Figure 9: images of M55 through the Bessel V filter opened through Astroart 8*

We can try to get a clearer image using the astropy library:

PHYS134L > test2.py > ...

```python
import os #This package allows you to interact with your operating system
import numpy as np #standard math library: https://numpy.org/doc/stable/
import matplotlib.pyplot as plt #standard plotting library: https://matplotlib.org/stable/index.html
from astropy.io import fits #Astropy is a multi-purpose python package made by astronomers: https://docs.astropy.org/en/stable/index.html
from astropy.stats import * #Astropy is massive and we only want specific tools from it for now, so it's best to import only what we need.
from photutils.background import Background2D, MedianBackground #Photutils is another package used to manipulate images and find sources in our images.
from photutils.detection import DAOStarFinder #Photutils documentation: https://photutils.readthedocs.io/en/stable/
from photutils.aperture import CircularAperture
from photutils.aperture import aperture_photometry

import warnings
warnings.filterwarnings('ignore')


image = fits.open ('/home/h4ckerman/PHYS134L/M55_fits/lsc0m409-kb98-20221108-0073-e91.fits')
image.info ()
image_data = image[0].data
image.close ()


plt.imshow(np.log(image_data), cmap='gray', vmin = 4.5, vmax = 5)
plt.colorbar ()

plt.show ()
```

*Figure 10: python code used to display the images*

We chose Vmin = 4.5 and Vmax = 5 for best visibility. For the Bessel B filter, the code gives us the image:

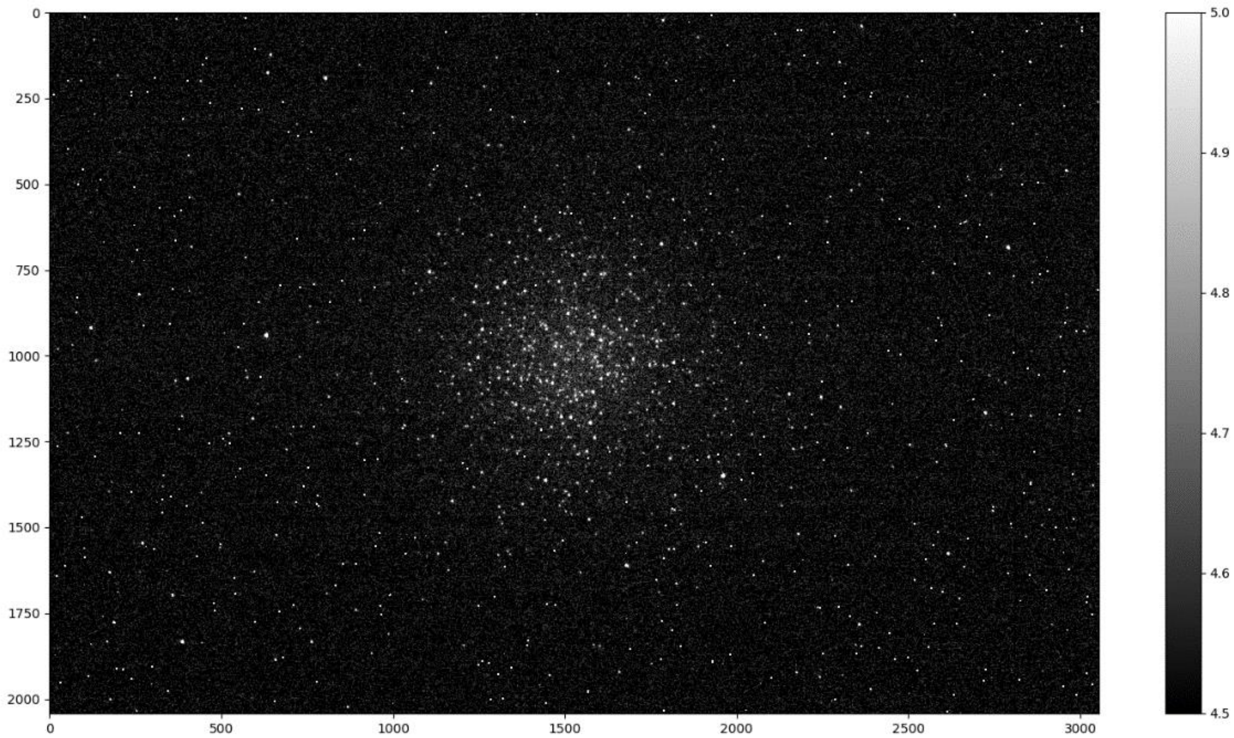



*Figure 11: images of M55 through the Bessel B filter opened through python*

Similarly for the Bessel V filter with Vmin = 5.5 and Vmax = 5.8, we get:

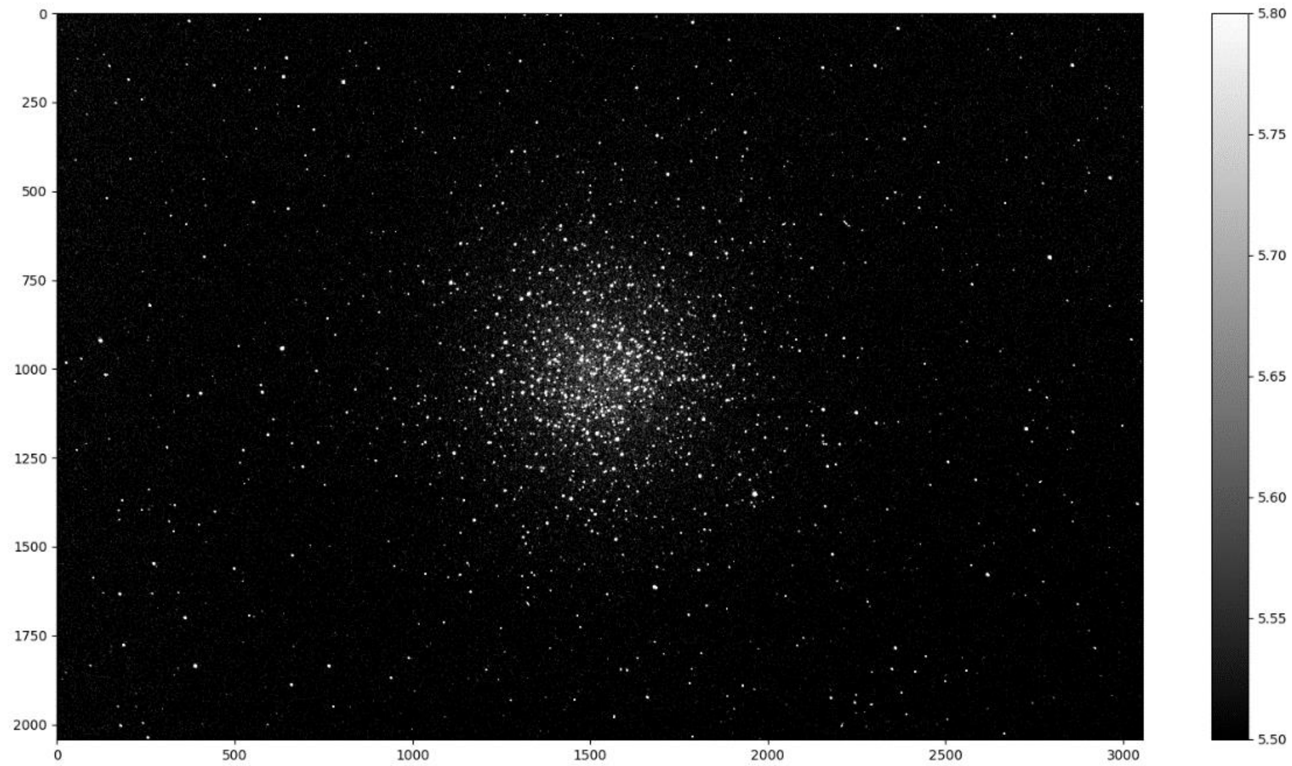



*Figure 12: images of M55 through the Bessel V filter opened through python*

We get images that look reasonably close to ESO's optical image. We can note that in both the Astroart images and the images opened in python, the image in the V filter looks brighter with more defined stars. This is to be expected as our stellar population has older and redder stars.

To see if the image is saturated, we can plot a histogram in both bands:

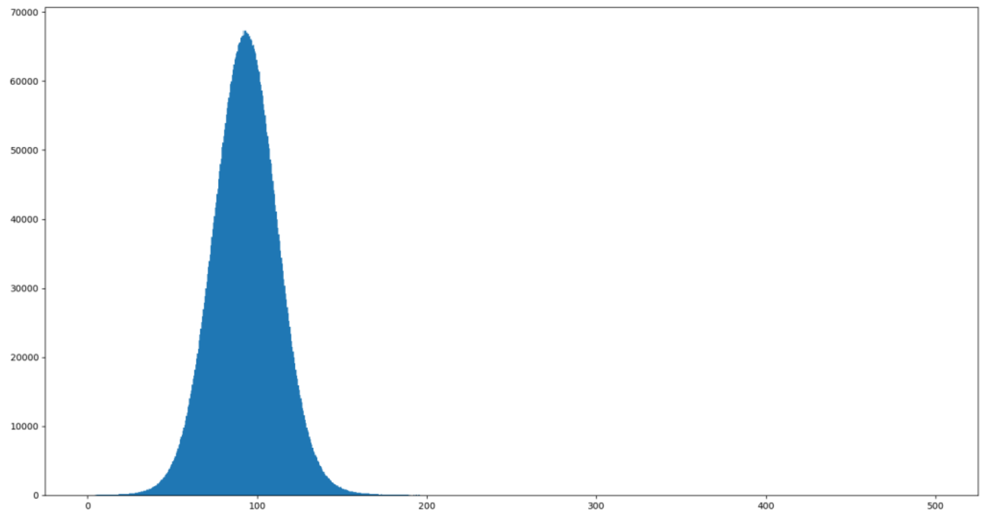



*Figure 13: Histogram for the B image*

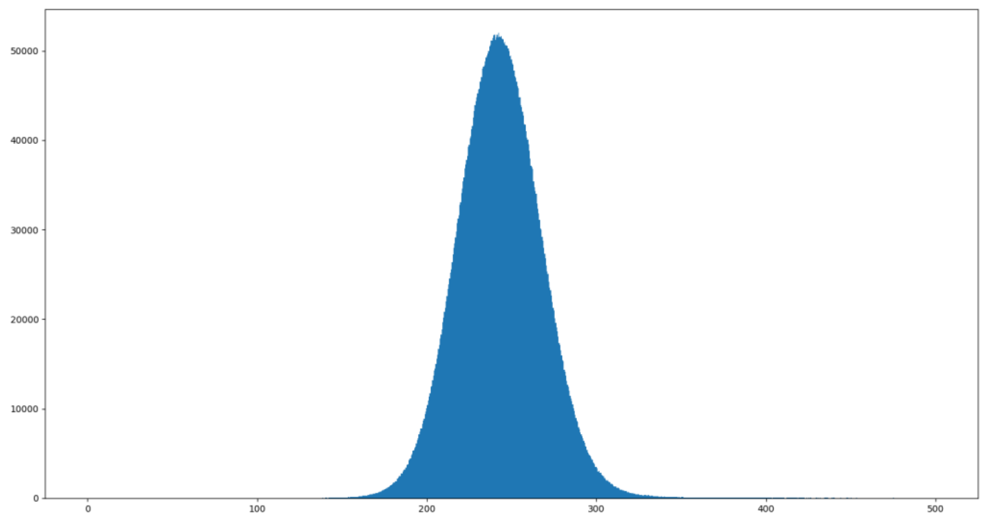



*Figure 14: Histogram for the V image*

We can see a smooth distribution; it seems the image is not saturated.

We can now use the photoutils module to try to find individual stars.

## C. Finding sources

To find the stars in our images, we used the photoutils library. There are two algorithms in photoutils that we can use to find sources: DAOstarfinder and IRAFstarfinder. DAOstarfinder is based on the DAOFIND algorithm developed by (Stetson, Peter, 1987). It tries to find stars by searching for local maxima in images who have a shape similar to a gaussian kernel with thresholds for amplitude of the maxima, sharpness and roundness. The IRAF algorithm is a similar algorithm developed by the National Optical Astronomy Observatories.

Both functions give us almost identical results with similar parameters. We will use the DAOstarfinder in our application as it has a little bit more documentation available.

Before trying to find the stars, we must isolate the background to subtract it from the image. For this, we use a sigma clip, we eliminate from the image pixels with a value over a multiple of the standard deviation. We can then eliminate the background from the image by subtracting it. This (in theory) leaves us with only the stars which we can find using the DAOstarfinder function.

For fwhm = 5 and threshold = 3.5*std, DAOstarfinder gives us 1524 sources for the B image and 2994 sources for the V image:

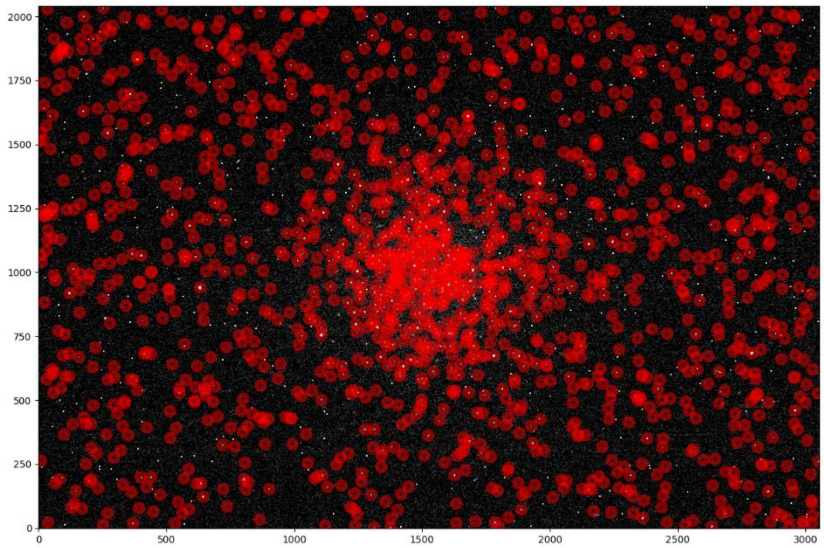



*Figure 15: Sources plotted over the B image*

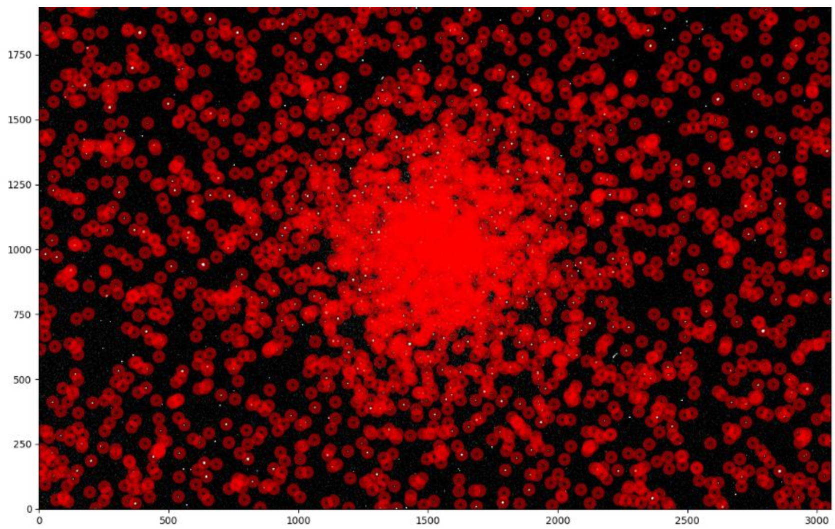



*Figure 16: sources plotted over the V image*

We can see that there is a good concentration of sources around the center of the image, as expected.

## D. Aligning the images

We can notice in that the images are not perfectly alligned by blinking them in astroart. To solve this alignement issue we tried several methods :

The first method we tried was using astroalign, a python library made specifically for aligning images that finds triangles of stars on the images and tries to find a linear transformation using those. Unfortunately this method wielded an esoteric error linked to the library pandas and from what we could tell, specifically it's distribution on linux and macOS.

The second method we tried to use to allign the images was using the calibration pixels located in the header of the fits files (under the name 'CRPIX1' and 'CRPIX2'). These are normally calibrated using Gaia's data but in our case, the calibration pixels in both images had the same value even though the images were clearly not alligned, making calibration using this unusable.

The third method we finally had to go through was to calibrate the images manually. For this we opened the images in astroart and tried to find stars that were clearly visible on both. We chose four stars for this process, one from each quadrant from the image. Thankfully on astroart the stars were less than 10 pixels away from each others meaning the images were not too misaligned. The DAOstarfinder function has a parameter "brightest" which makes the function return the n brightest sources in an ordered way. It also has a "mask" parameter which takes into input a numpy ndarry with the same size as the images with boolean values to indicate wether a zone is masked or not (0 for no mask, 1 for mask). We used these two parameters to make a mask over the entire image except a 20*20 zone around the estimated coordinates of the reference stars and find the 4 brigthest stars in the masked image (hopefully returning us only the 4 reference stars).

```python
def find_deltas (image_data_B, image_data_V):
    mask_calibration = np.ones (image_data_V.shape, dtype = bool)

    mask_default = np.zeros (image_data_B.shape, dtype = bool)
    mask_calibration[1810:1850, 370:410] = False
    mask_calibration[925:965, 615:655] = False
    mask_calibration[665:705, 2770:2810] = False
    mask_calibration[1560:1600, 2600:2640] = False

    sources_B = find_sources_DAO (image_data_B, 5, 3.5, 2, mask = mask_calibration, nb = 4)
    sources_V = find_sources_DAO (image_data_V, 5, 3.5, 2, mask = mask_calibration, nb = 4)

    reference_list_B = [[x['xcentroid'] for x in sources_B],[y['ycentroid'] for y in sources_B]]
    reference_list_V = [[x['xcentroid'] for x in sources_V],[y['ycentroid'] for y in sources_V]]

    delta_x_list = [reference_list_B[0][i] - reference_list_V[0][i] for i in range(len(reference_list_B[0]))]
    delta_y_list = [reference_list_B[1][i] - reference_list_V[1][i] for i in range(len(reference_list_B[1]))]
    delta_x = np.mean (delta_x_list)
    delta_y = np.mean (delta_y_list)

    return (delta_x, delta_y)
```

*Figure 17: code for the calibration*

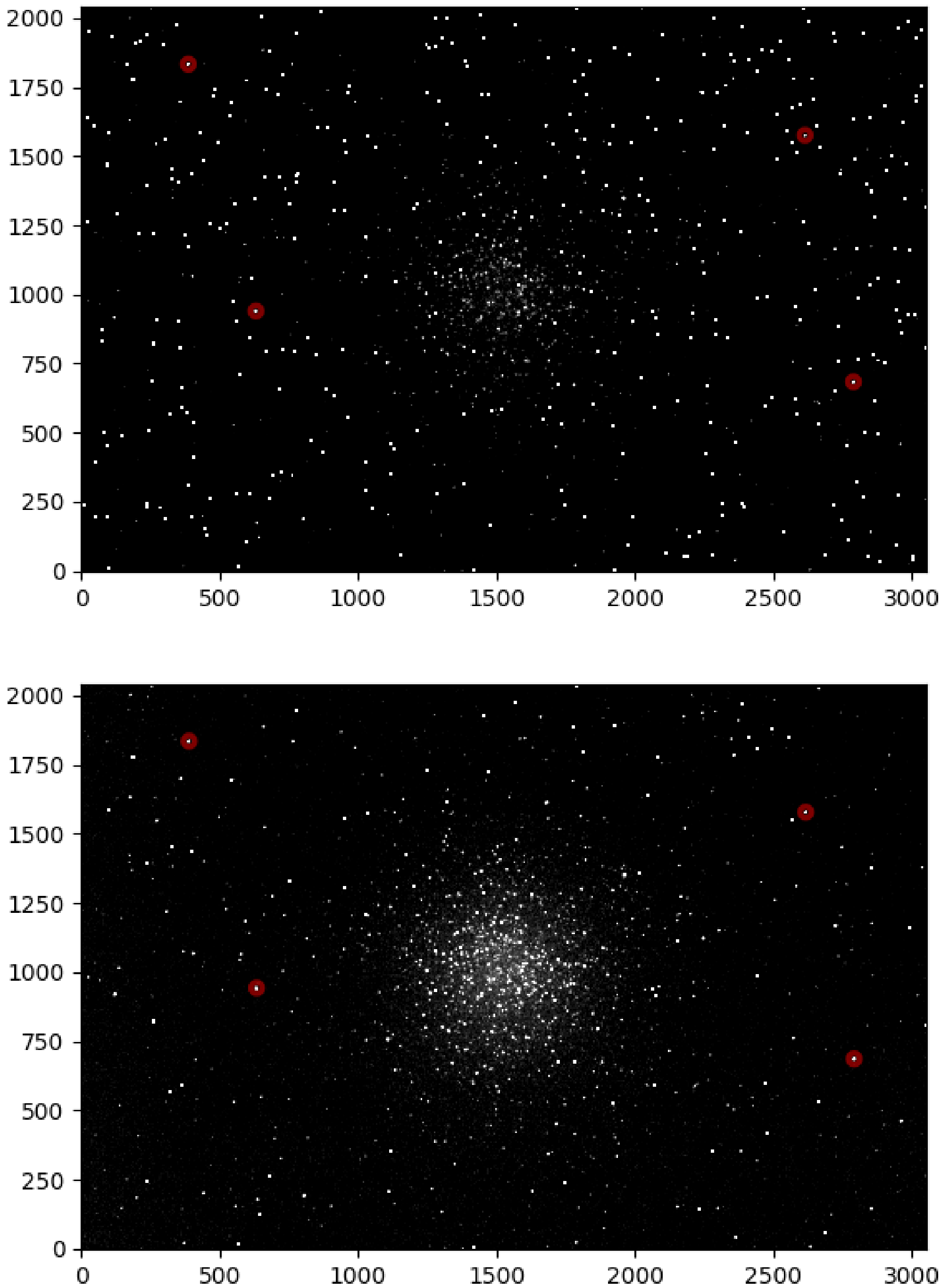



*Figure 18: Reference stars plotted over the B and V images*

We can see that in the above images, the stars found in both images are the same. We then average out the x and y pixel values for the stars to find an offset in the x and y axes between the images which we can add to the x and y values of the V image sources.

## E. Matching stars

Now that we have two lists for the sources found in the B image and in the V image, we now need to match these sources.

```
def find_close_sources (sourceA, sourceB, threshold):
    n = 0
    sources_list1 = []
    sources_list2 = []
    V_mag_list = []
    mag_list = []

    threshold = float (threshold)
    for i in sourceA :
        for j in sourceB :
            if ((np.abs(i['xcentroid'] - j['xcentroid']) <=  threshold) and (np.abs (i['ycentroid'] - j['ycentroid']) <= threshold)) :
                sources_list1.append (i)
                sources_list1.append (j)

                n += 1
                mag_list.append(i['mag'] - j['mag'])
                V_mag_list.append (j['mag'])

    return (n, sources_list1, sources_list2, mag_list, V_mag_list)
```

*Figure 19: code used to find close sources*

For this we define a certain threshold for the closeness of the x and y positions to find matching stars. For around 1700 stars in the B image and 3000 in the V image, we found around 400 matches.

Now that we have B-V and V magnitude values for matching stars, we can plot our H-R diagram.

# IV. M55's H-R diagram

```
delta_x, delta_y = find_deltas (image_data_B, image_data_V)

mask_default = np.zeros (image_data_B.shape, dtype = bool)

sources_B = find_sources_DAO (image_data_B, 5, 3.5, 2, mask = mask_default, nb = 5000)
sources_V = find_sources_DAO (image_data_V, 5, 3.5, 2, mask = mask_default, nb = 5000)

for i in sources_V:
    i['xcentroid'] += delta_x
    i['ycentroid'] += delta_y

print (sources_B, sources_V)

n, l1, l2, B_V_list, V_mag_list = find_close_sources (sources_B, sources_V, 0.5)
print (n)

plt.scatter (B_V_list, V_mag_list, marker='+')
plt.gca().invert_yaxis()
plt.title ("H-R diagram of M55")
plt.ylabel ("V magnitude")
plt.xlabel ('B-V')

plt.show()
```

*Figure 20: Code used to plot the H-R diagram*

Using the code above, we can plot M55's H-R diagram:

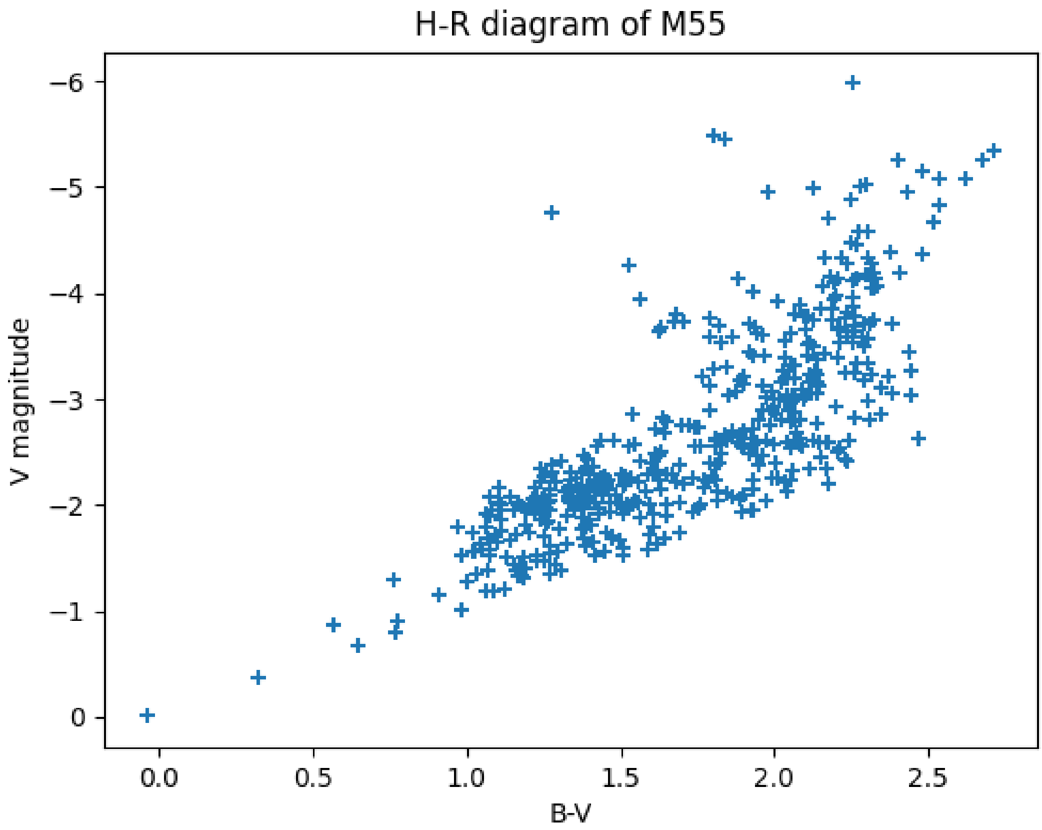



*Figure 21: M55's H-R diagram*

Please note that the magnitudes are not calibrated as we couldn't find reference stars to compare magnitudes on SIMBAD. Instead, they are calculated using the expression:

$$m = -2.5\log\left(\frac{peak\ density}{detection\ threshold}\right)$$

This might cause the B-V index to be wrong and our H-R diagram to have the wrong shape. Nevertheless, we can compare this H-R diagram to previously done diagrams, for instance Alcaino et al. (1992) and Zaggia et al. (1997).

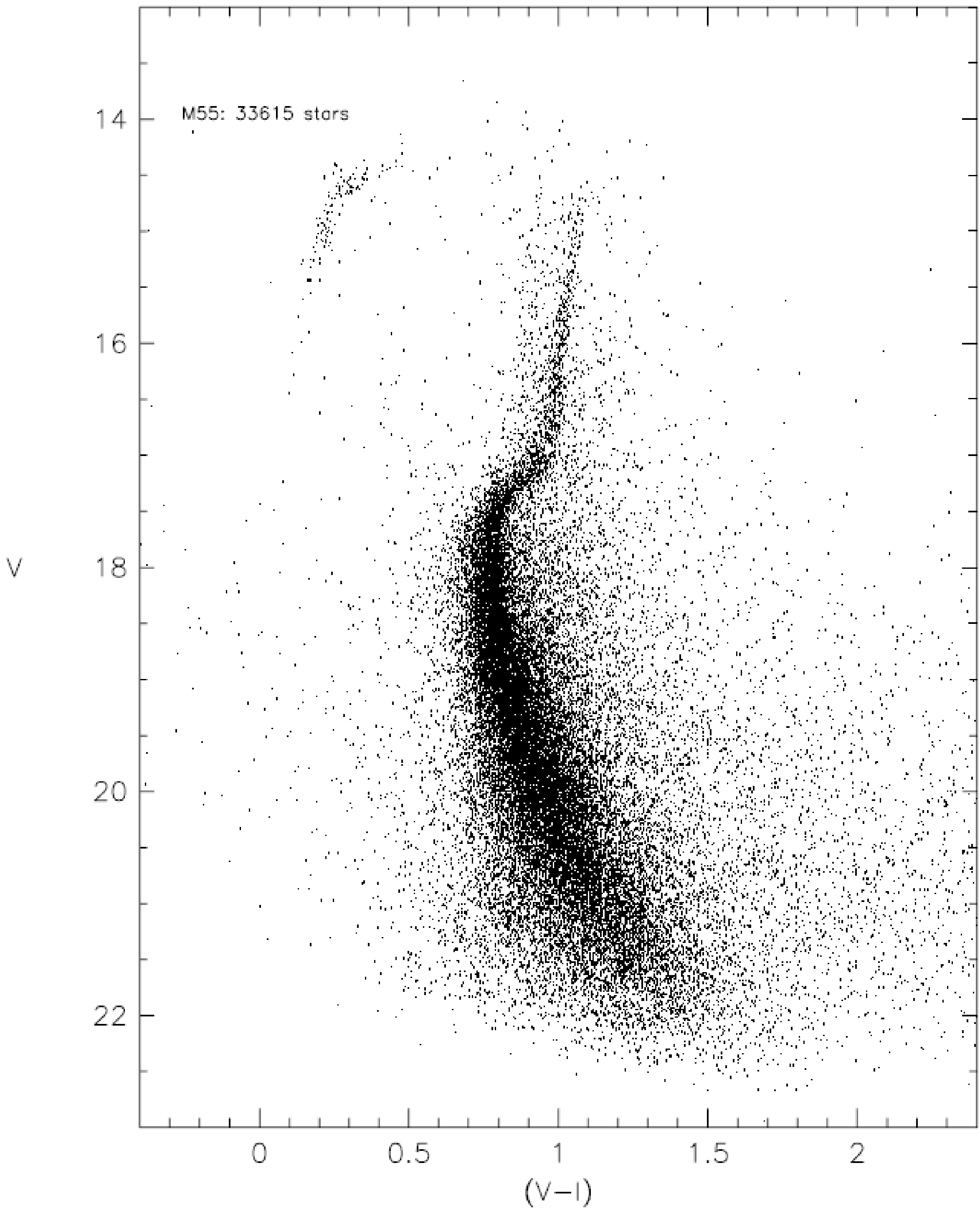
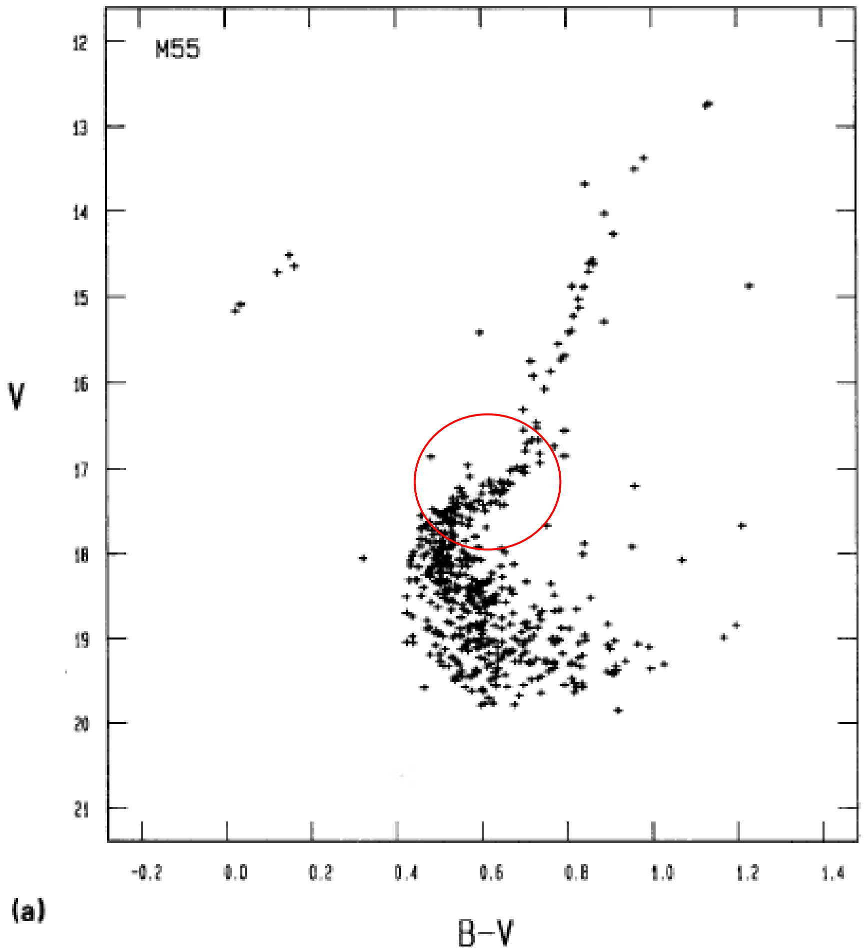



*Figure 22: Color index diagram of M55 made by Zaggia et al. (1997)*




It is reasonable to think that we won't have any stars bellow magnitude ~19 as they are too dim to be detected by our star finding functions. Given this information, we think our H-R diagram represents the zone circled in red.

It is interesting to note that both in the 1992 H-R diagram and in ours, we see little evidence of the blue straggler stars that are present in the NASA image bellow.

*Figure 23: H-R diagram of M55 by Alcaino et al. (1992)*

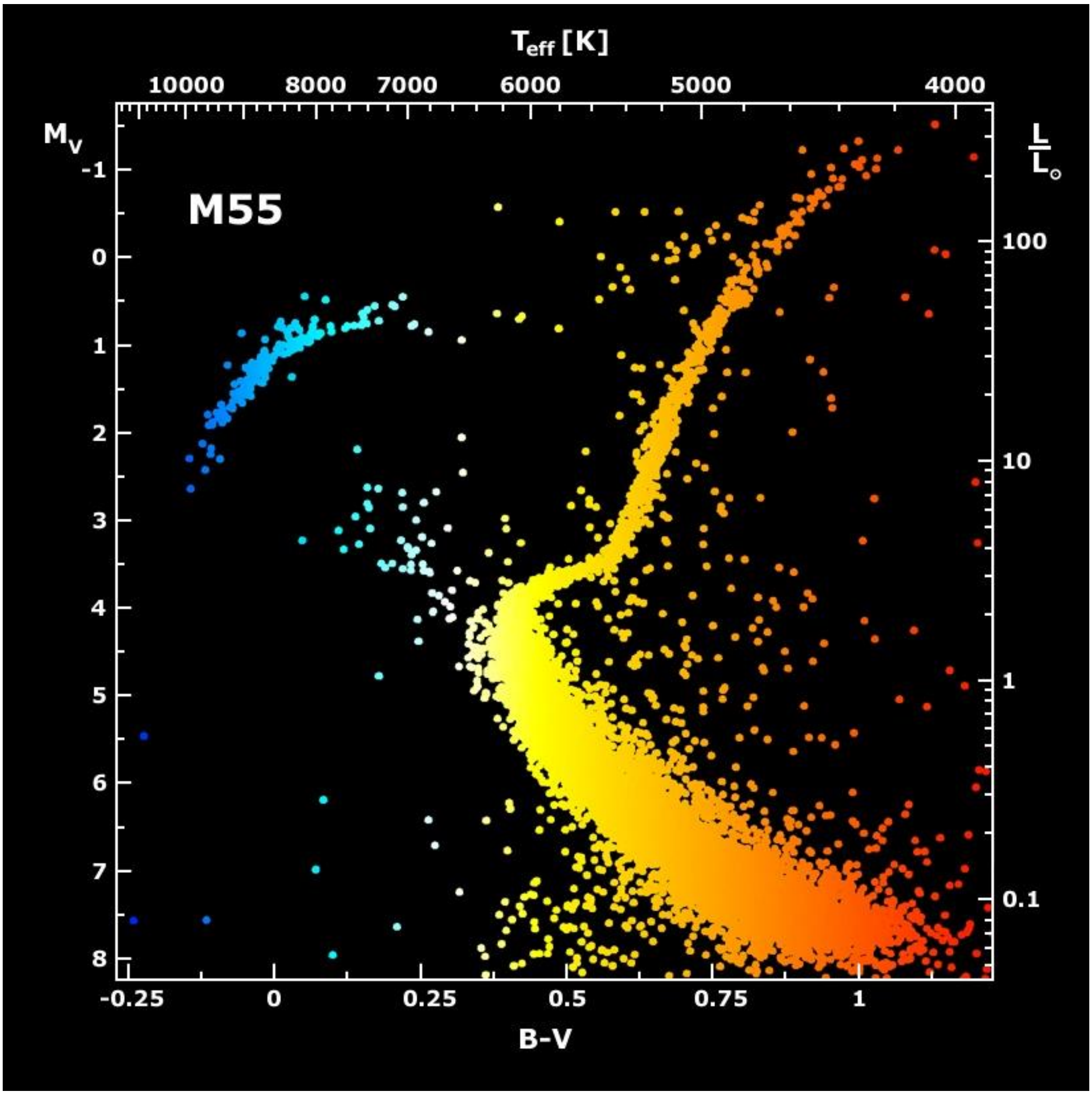



*Figure 24: H-R diagram of M55 as shown on the APOD of February 23 2001*

## V. Conclusion and if we had to do this again

In conclusion, we used LCO's 0.4 m telescope to get two images of M55, one in the B band and one in the V band with integration times t=100s. We then used the astropy and photutils python libraries to find stars and match them. Using these matched stars, we plotted their V magnitudes over their B-V to get M55's H-R diagram.

If we had to do it all again, we would maybe try to find reference stars to calibrate our B and V magnitudes to get a better H-R diagram. We would also try to use multiple images transposed to further reduce the background noise.

# References

Alcaino, L. A. (1992). BVRI CCD photometry of the globular cluster M55 (NGC 6809). *The Astronomical Journal*.

*APOD Feb 23 2001*. (2001, Feburary 23). Retrieved from https://apod.nasa.gov/apod/ap010223.html.

Barnes, J. (1993). A Beginner's Guide to Using IRAF. *National optical astronomy observatories*.

Baumgardt, H. (2018, April 23). A catalogue of masses, structural parameters, and velocity dispersion of 112 Milky Way globular clusters. *Monthly Notices of the Royal Astronomical Society*.

Beroiz, C. S. (2020). Astroalign: A Python module for astronomical image registration. *Astronomy and computing*.

Ferraro, L. R. (2018). The Hubble Space Telescope UV Legacy Survey of Galactic Globular Clusters - XV. The dynamical clock: reading cluster evolution from the segregation of blue straggler stars. *Astrophysics of Galaxies*.

Iraf. (n.d.). *Iraf.net.* Retrieved from https://iraf.net/.

L.Collina, R. B. (1996). Absolute Flux Calibrated Spectrum of Vega. *Instrument Science Report CAL/SCS-008*.

Lee, S.-W. (1977). The C-M diagram of the globular cluster M55. *Astronomy and Astrophysics Supplement Series*.

Lenoard, P. (1989). Stellar Collisions in Globular Clusters and the Blue Straggler Problem . *Astronomical Journal*.

Shapely. (1918). Globular clusters and the structure of the galactic system. *The Astronomical Society of the Pacific*.

Shapely, S. (1927). A classification of globular clusters. *Harvard College Observatory Bulletin*.

Stetson, P. (1987). DAOPHOT: A Computer Program for Crowded-Field Stellar Photometry. *Publications of the Astronomical Society of the Pacific*.

Zaggia, P. C. (n.d.). 1997. *Astronomy and Astrophysics*.

# Appendix A: code for the H-R diagram

```python
import os #This package allows you to interact with your operating system
import numpy as np #standard math library: https://numpy.org/doc/stable/
import matplotlib.pyplot as plt #standard plotting library: https://matplotlib.org/stable/index.html
from astropy.io import fits #Astropy is a multi-purpose python package made by astronomers: https://docs.astropy.org/en/stable/index.html
from astropy.stats import * #Astropy is massive and we only want specific tools from it for now, so it's best to import only what we need.
from photutils.background import Background2D, MedianBackground #Photutils is another package used to manipulate images and find sources_B in our images.
from photutils.detection import DAOStarFinder #Photutils documentation: https://photutils.readthedocs.io/en/stable/
from photutils.detection import IRAFStarFinder
from photutils.aperture import CircularAperture
from photutils.aperture import aperture_photometry
import astroalign as aa
import warnings
warnings.filterwarnings('ignore')

def magnitude_to_flux (m, filter = 'none'): #Vega magnitude
    reference_flux = [2.1*(10**7), 3.18*(10**7), 1.82*(10**7), 1.10*(10**7), 5.66*(10**6)] #data from Bessell et al. (1998)
    reference_mag = [0.030, 0.035, 0.035, 0.075, 0.095]# (Johnson, 1964, 1965) average
    if filter == "none" :
        filter = str( input ("Which filter ? (U,B,V,R,I) :"))
    list_filters = ["U","B","V","R","I"]
    filter_index = list_filters.index (list(set(list_filters).intersection(set(filter)))[0])
    m_vega = reference_mag [filter_index]
    flux_vega = reference_flux [filter_index]
    flux = np.power (10, 0.4*((m_vega) - m))*flux_vega
    return (flux)

def get_data_header (image_path): #gets the data and header from the fits file
    image = fits.open (image_path)
    primary_hdu = image['PRIMARY'].data
    header = image['PRIMARY'].header
    image.close ()
    return (primary_hdu, header)

def get_hdu_stats (image_data): #gets stats
    min = np.min (image_data)
    max = np.max (image_data)
    mean = np.mean (image_data)
    std = np.std (image_data)
    return (min, max , mean, std)

def plot_histogram (image_data):
    image_data_flat = image_data.flatten ()
    histogram = plt.hist (image_data_flat, bins = np.linspace(0,500,1000))
    plt.show ()

def get_background_median (image_data):
    sigma_clip = SigmaClip(sigma=3.0, maxiters= 5)
    bkg_estimator = MedianBackground()
    background = Background2D (image_data, box_size= (50,50), filter_size = (3,3), sigma_clip= sigma_clip, bkg_estimator= bkg_estimator)
    return (background.background_median)

def find_sources_DAO (image_data, FWHM, thresh, sharphi, mask, nb):
    mean, median, std = sigma_clipped_stats(image_data, sigma=3.0)
    daofind = DAOStarFinder (fwhm= FWHM, threshold= thresh*std, sharphi= sharphi, brightest= nb)
    sources_B = daofind (image_data - median, mask = mask)
    return (sources_B)

def find_sources_DAO_invert (image_data, FWHM, thresh, sharphi, mask):
    mean, median, std = sigma_clipped_stats(image_data, sigma=3.0)
    daofind = DAOStarFinder (fwhm= FWHM, threshold= thresh*std, sharphi= sharphi)
    sources_B = daofind (image_data - median, mask = mask)
    try :
        shape = image_data.shape
        for i in sources_B:
            i['xcentroid'] = shape[1] - i['xcentroid']
            i['ycentroid'] = shape[0] - i['ycentroid']
    except :
        print ("errorrrrr")
    return (sources_B)
```

```python
def find_sources_IRAF (image_data, FWHM, thresh, sharphi):
    mean, median, std = sigma_clipped_stats(image_data, sigma=3.0)
    irafind = IRAFStarFinder (fwhm= FWHM, threshold= thresh*std, sharphi= sharphi)
    sources_B = irafind (image_data - median)
    return (sources_B)

def show_image (image_data, vmin, vmax):
    plt.imshow(np.log(image_data), cmap='gray', vmin = vmin, vmax = vmax)
    plt.colorbar ()
    plt.show ()

def plot_sources (image_data, sources_B, vmin, vmax):
    positions = np.transpose((sources_B['xcentroid'], sources_B['ycentroid']))
    apertures = CircularAperture(positions, r=10)
    plt.figure(figsize=(10,10))

    plt.imshow(np.log(image_data), origin='lower', cmap = "gray", vmin=vmin, vmax = vmax)
    v = apertures.plot(color='red', lw=5, alpha=0.5)
    plt.show()

def find_deltas (image_data_B, image_data_V):
    mask_calibration = np.ones (image_data_V.shape, dtype = bool)

    mask_default = np.zeros (image_data_B.shape, dtype = bool)
    mask_calibration[1810:1850, 370:410] = False
    mask_calibration[925:965, 615:655] = False
    mask_calibration[665:705, 2770:2810] = False
    mask_calibration[1560:1600, 2600:2640] = False

    sources_B = find_sources_DAO (image_data_B, 5, 3.5, 2, mask = mask_calibration, nb = 4)
    sources_V = find_sources_DAO (image_data_V, 5, 3.5, 2, mask = mask_calibration, nb = 4)

    reference_list_B = [[x['xcentroid'] for x in sources_B],[y['ycentroid'] for y in sources_B]]
    reference_list_V = [[x['xcentroid'] for x in sources_V],[y['ycentroid'] for y in sources_V]]
    delta_x_list = [reference_list_B[0][i] - reference_list_V[0][i] for i in range(len(reference_list_B[0]))]
    delta_y_list = [reference_list_B[1][i] - reference_list_V[1][i] for i in range(len(reference_list_B[1]))]
    delta_x = np.mean (delta_x_list)
    delta_y = np.mean (delta_y_list)

    return (delta_x, delta_y)

def find_close_sources (sourceA, sourceB, threshold):
    n = 0
    sources_list1 = []
    sources_list2 = []
    V_mag_list = []
    mag_list = []

    threshold = float (threshold)
    for i in sourceA :
        for j in sourceB :
            if ((np.abs(i['xcentroid'] - j['xcentroid']) <=  threshold) and (np.abs (i['ycentroid'] - j['ycentroid']) <= threshold)) :
                sources_list1.append (i)
                sources_list1.append (j)

                n += 1
                mag_list.append(i['mag'] - j['mag'])
                V_mag_list.append (j['mag'])

    return (n, sources_list1, sources_list2, mag_list, V_mag_list)


try:
    image_data_B, header_B = get_data_header ('/home/h4ckerman/PHYS134L/M55_fits/lsc0m409-kb98-20221108-0073-e91.fits')
    image_data_V, header_V = get_data_header ('/home/h4ckerman/PHYS134L/M55_fits/lsc0m409-kb98-20221108-0074-e91.fits')
except :
    image_data_B, header_B = get_data_header ('lsc0m409-kb98-20221108-0073-e91.fits')
    image_data_V, header_V = get_data_header ('lsc0m409-kb98-20221108-0073-e91.fits')
#image_data_B, align_footprint = aa.register (image_data_B, image_data_V, detection_sigma=2)


delta_x, delta_y = find_deltas (image_data_B, image_data_V)

mask_default = np.zeros (image_data_B.shape, dtype = bool)

sources_B = find_sources_DAO (image_data_B, 5, 3.5, 2, mask = mask_default, nb = 5000)
sources_V = find_sources_DAO (image_data_V, 5, 3.5, 2, mask = mask_default, nb = 5000)

for i in sources_V:
    i['xcentroid'] += delta_x
    i['ycentroid'] += delta_y

print (sources_B, sources_V)

n, l1, l2, B_V_list, V_mag_list = find_close_sources (sources_B, sources_V, 0.5)
print (n)

plt.scatter (B_V_list, V_mag_list, marker='+')
plt.gca().invert_yaxis()
plt.title ("H-R diagram of M55")
plt.ylabel ("V magnitude")
plt.xlabel ('B-V')

plt.show()
```

# Appendix B: code for the S/N ratio calculations and plot

```python
import numpy as np
import tkinter as tk
import matplotlib.pyplot as plt

def calculation (F,Fsky, Nr, Idc, Qe, Ps, Eff, t, A): #based on UCSB PHYS134L notes
    Ae  = A*Eff*Qe
    SN = (F*Ae*np.sqrt(t))/np.sqrt((np.power(Nr,2)/t) + F*Ae + Idc + Fsky*Ae*Ps)
    return (SN)

def var_input ():
    print ("please input the following and press enter :")
    F = float (input ("Point Source Signal Flux on Telescope (in pixel/s*cm^2 :"))
    Fsky = float (input ("Background Flux from Sky (in pixel/s*cm^2*arcsecond^2 :"))
    Nr = float (input ("Readout noise :"))
    Idc = float (input ("Dark current (in e-/s :"))
    Qe = float (input ("Qantum efficency :"))
    Ps = float (input ("Pixel size (arcsec) :"))
    Eff = float (input ("Efficency :"))
    t = float (input ("integration time (s) :"))
    A = float (input ("Telescope area (cm^2) :"))
    return (F,Fsky,Nr,Idc,Qe,Ps,Eff,t,A)

def magnitude_to_flux (m, filter): #Vega magnitude
    reference_flux = [756.1, 1392.6, 995.5, 702.0, 452.0]
    reference_mag = [0.030, 0.035, 0.035, 0.075, 0.095]# (Johnson, 1964, 1965) average
    if filter == "none" :
        filter = str( input ("Which filter ? (U,B,V,R,I) :"))
    list_filters = ["U","B","V","R","I"]
    filter_index = list_filters.index (list(set(list_filters).intersection(set(filter)))[0])
    m_vega = reference_mag [filter_index]
    flux_vega = reference_flux [filter_index]
    flux = np.power (10, 0.4*(m_vega - m))*flux_vega
    return (flux)

def magnitude_to_flux_2 (m, filter = 'none'): #Vega magnitude
    reference_flux = [2.1*(10**7), 3.18*(10**7), 1.82*(10**7), 1.10*(10**7), 5.66*(10**6)] #data from Colina et al. (1997)
    reference_mag = [0.030, 0.035, 0.035, 0.075, 0.095]
    if filter == "none" :
        filter = str( input ("Which filter ? (U,B,V,R,I) :"))
    list_filters = ["U","B","V","R","I"]
    filter_index = list_filters.index (list(set(list_filters).intersection(set(filter)))[0])
    m_vega = reference_mag [filter_index]
    flux_vega = reference_flux [filter_index]
    flux = np.power (10, 0.4*((m_vega) - m))*flux_vega
    return (flux)

def magnitude_to_flux_example (m, filter = "none"): #Vega magnitude
    m_example = 20
    flux_example = 0.03
    flux = np.power (10, 0.4*(m_example - m))*flux_example
    return (flux)

def dark_current (idc, t):
    return (idc*t)

def background_noise (Fsky, A, Eff, Qe, Ps, t):
    return (Fsky*A*Eff*Qe*Ps*t)

def signal (F,t,A,Eff,Qe):
    return (F*t*A*Eff*Qe)

#print (calculation (*var_input()))
def plot_graph ():

    magnitudes_list = np.linspace (5, 20, 4)
    integration_list = np.linspace (1, 1001, 21) #increments of 50


    SNlist = []
    for m in magnitudes_list :
        t_list = []
        F = magnitude_to_flux_2 (m, 'B')
```

```python
        Fsky = magnitude_to_flux_2 (22, 'B')

        for t in integration_list :
            t_list.append (calculation (F, Fsky, 14.5, 0.03, 0.3,  0.571, 0.3, t, 25))
        SNlist.append (t_list)

    plt.plot (np.log (integration_list), np.log(SNlist[0]), color = "red")
    plt.plot (np.log (integration_list), np.log(SNlist[1]), color = "orange")
    plt.plot (np.log (integration_list), np.log(SNlist[2]), color = "yellow")
    plt.plot (np.log (integration_list), np.log(SNlist[3]), color = "green")
    plt.xlabel ("log of integration time")
    plt.ylabel ("log of S/N ratio")
    plt.title ("S/N ratio over integration time for different magnitudes")
    plt.legend (["m=5", "m=10", "m=15", "m=20"])

    plt.show ()


    dark_current_list = []
    background_noise_list = []
    readout_noise_list = []
    signal_list = []

    for t in integration_list :
        dark_current_list.append (dark_current(1,t))
        background_noise_list.append (background_noise(np.power(10.0,-2), 1000, 0.5, 0.3, 4, t))
        readout_noise_list.append (12)
        signal_list.append (signal (0.03, t, 1000, 0.5, 0.3))


    all_noise_list = [sum(i) for i in zip(dark_current_list, background_noise_list, readout_noise_list)]
    print (len(all_noise_list))
    plt.plot (integration_list, dark_current_list, integration_list, background_noise_list, integration_list, readout_noise_list)
    plt.plot (integration_list, all_noise_list, color = "blue")
    plt.plot (integration_list, signal_list, color = "red")

    plt.show ()


def plot_graph2 ():

    magnitudes_list = np.linspace (10, 20, 100)
    t = 100

    SNlist_B = []
    for m in magnitudes_list :
        F = magnitude_to_flux_2 (m,'B')
        Fsky = magnitude_to_flux_2 (22, 'B')
        SNlist_B.append (calculation (F, Fsky, 14.5, 0.03, 0.3,  0.571, 0.3, t, 25))

    plt.plot (magnitudes_list, SNlist_B, color = "blue")
    plt.xlabel ("magnitude")
    plt.ylabel ("S/N ratio")
    plt.title ("S/N ratio over magnitudes in the B band")

    plt.show ()

    SNlist_V = []
    for m in magnitudes_list :
        F = magnitude_to_flux_2 (m,'V')
        Fsky = magnitude_to_flux_2 (22, 'V')
        SNlist_V.append (calculation (F, Fsky, 14.5, 0.03, 0.3,  0.571, 0.3, t, 25))

    plt.plot (magnitudes_list, SNlist_V, color = "green")
    plt.xlabel ("magnitude")
    plt.ylabel ("S/N ratio")
    plt.title ("S/N ratio over magnitudes in the V band")

    plt.show ()

plot_graph ()
plot_graph2 ()
```