Lab 2 Report

Jack Colvin, Along with my group, Larrence Xing, Kevin Wu, Annabelle Yarborough and Olivia Lucal

Part One: Creating RGB Images!

In this part of the lab, I used a combination of our 4 observations (M87, M96, NGC 3227, NGC 3156), and 6 archival galaxies (m51, m64, m66, ngc3180, ngc2095, and ngc3941) to create RGB images, using the i-band as our red color, r-band as our green color, and g-band as our blue color, just as we did for the previous lab. In making the images, I aimed for two things: I wanted the color of the star to be white or yellow, and the background to be dark without any one color dominating. To acheive these two things, I used z-scale on each of the frames to make them visible, and did further scaling on each of the individual color to make sure they were balanced in the images. I added some additional archival galaxies in order to see if I could get a closer value for the hubble constant, with mixed results, and beacuse I was enjoying the lab and thought it would be interesting to try with a larger sample size to see if my trend still held.

Here are the links to the observer folders containing my archival galaxies:

m66, ngc 2095, ngc 3941:
https://stars.uchicago.edu/images/StoneEdge/0.5meter/2025/2025-05-01/sophiawinney/

ngc 3180: https://stars.uchicago.edu/images/StoneEdge/0.5meter/2025/2025-04-29/evasch1/

m64: https://stars.uchicago.edu/images/StoneEdge/0.5meter/2025/2025-04-18/ahimelhoch/

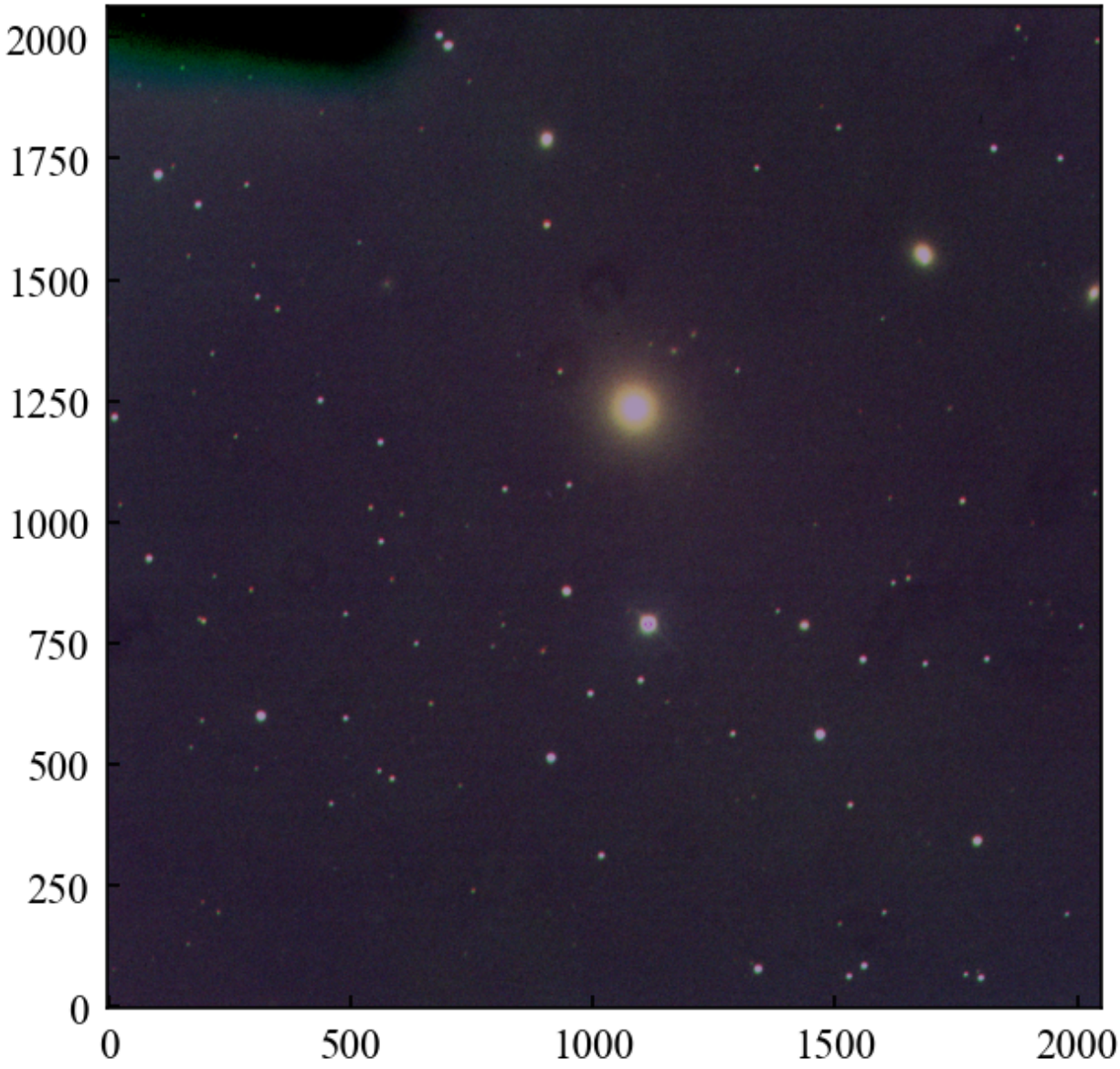
m51: https://stars.uchicago.edu/images/StoneEdge/0.5meter/2025/2025-04-22/klyachman/

In [58]:
```
import numpy as np
from astropy.io import fits
import matplotlib.pyplot as plt
from astropy.visualization import ZScaleInterval
from matplotlib.colors import LogNorm
from astropy.nddata import Cutout2D
from astropy.modeling import models, fitting
from astropy.visualization.lupton_rgb import make_lupton_rgb
from astropy.visualization import LogStretch
from PIL import Image

def plot_prettier(dpi=150, fontsize=11, usetex=False):
    '''
    Make plots look nicer compared to Matplotlib defaults
    Parameters:
        dpi - int, "dots per inch" - controls resolution of PNG images that
                by Matplotlib
        fontsize - int, font size to use overall
        usetex - bool, whether to use LaTeX to render fonds of axes labels
                use False if you don't have LaTeX installed on your system
    '''
    plt.rcParams['figure.dpi']= dpi
    plt.rc("savefig", dpi=dpi)
    plt.rc('font', size=fontsize)
    plt.rc('xtick', direction='in')
    plt.rc('ytick', direction='in')
    plt.rc('xtick.major', pad=5)
    plt.rc('xtick.minor', pad=5)
    plt.rc('ytick.major', pad=5)
    plt.rc('ytick.minor', pad=5)
    plt.rc('lines', dotted_pattern = [2., 2.])
    if usetex:
        plt.rc('text', usetex=usetex)
    else:
        plt.rcParams['mathtext.fontset'] = 'cm'
        plt.rcParams['font.family'] = 'serif'
        plt.rcParams['font.serif'] = ['Times New Roman'] + plt.rcParams['fon


plot_prettier(dpi=150, fontsize=11)
```

M87

In [59]:
```
iband = fits.open("/Users/jackcolvin//m87_i-band_120.0s_bin1_250429_055923_j
gband = fits.open("/Users/jackcolvin//m87_g-band_120.0s_bin1_250429_055639_j
rband = fits.open("/Users/jackcolvin//m87_r-band_120.0s_bin1_250429_055407_j
```

In [60]:
```
from astropy.visualization.lupton_rgb import make_lupton_rgb
from astropy.visualization import LogStretch


r = iband[0].data
g = rband[0].data
b = gband[0].data

#lupton_rgb doesn't support scaling
#so that must be done manually
zscale = ZScaleInterval()
r_scaled = zscale(r)
g_scaled = zscale(g)
b_scaled = zscale(b)



image = make_lupton_rgb(.95*r_scaled, .87*g_scaled, b_scaled, Q = 0, stretch
plt.imshow(image, origin = 'lower')
plt.show()
```

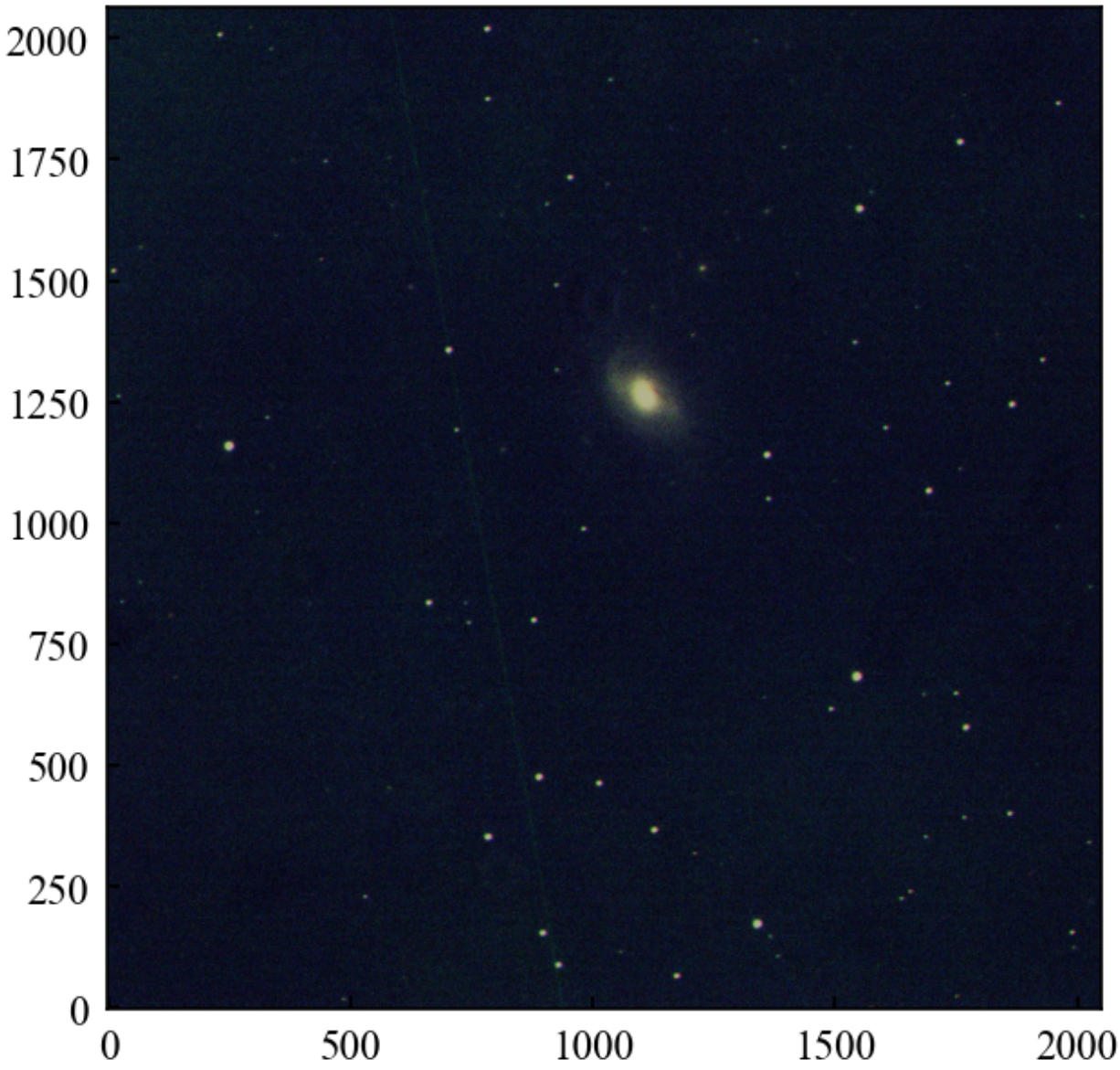


In [61]:
```
iband.close()
rband.close()
gband.close()
```

M96

In [62]:
```
iband1 = fits.open("/Users/jackcolvin//m96_i-band_60.0s_bin1_250429_053356_j
gband1 = fits.open("/Users/jackcolvin//m96_g-band_60.0s_bin1_250429_053017_j
rband1 = fits.open("/Users/jackcolvin//m96_r-band_60.0s_bin1_250429_053527_j
```

In [63]:
```
r1 = iband1[0].data
g1 = rband1[0].data
b1 = gband1[0].data

#lupton_rgb doesn't support scaling
#so that must be done manually
zscale = ZScaleInterval()
r_scaled = zscale(r1)
g_scaled = zscale(g1)
b_scaled = zscale(b1)



image = make_lupton_rgb(r_scaled, g_scaled, .85*b_scaled, Q = 0, stretch = 1
plt.imshow(image, origin = 'lower')
plt.show()
```

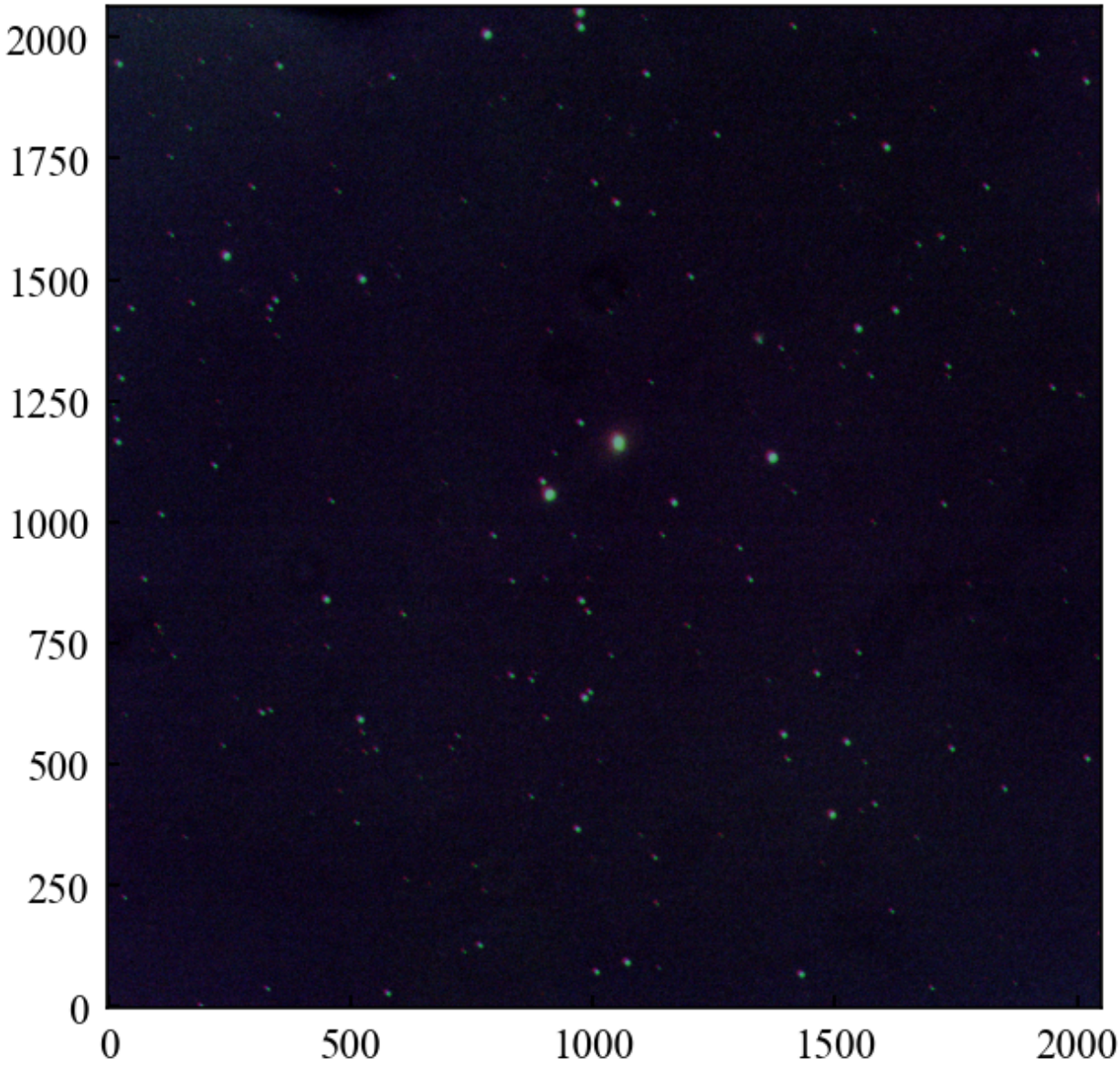


In [64]:
```
iband1.close()
rband1.close()
gband1.close()
```

NGC 3516

In [65]:
```
iband2 = fits.open("/Users/jackcolvin//ngc3516_i-band_120.0s_bin1_250429_054
gband2 = fits.open("/Users/jackcolvin//ngc3516_g-band_120.0s_bin1_250429_054
rband2 = fits.open("/Users/jackcolvin//ngc3516_r-band_120.0s_bin1_250429_054
```

In [66]:
```
r = iband2[0].data
g = rband2[0].data
b = gband2[0].data

#lupton_rgb doesn't support scaling
#so that must be done manually
zscale = ZScaleInterval()
r_scaled = zscale(r)
g_scaled = zscale(g)
b_scaled = zscale(b)



image = make_lupton_rgb(.75*r_scaled, g_scaled, .9*b_scaled, Q = 0, stretch
plt.imshow(image, origin = 'lower')
plt.show()
```

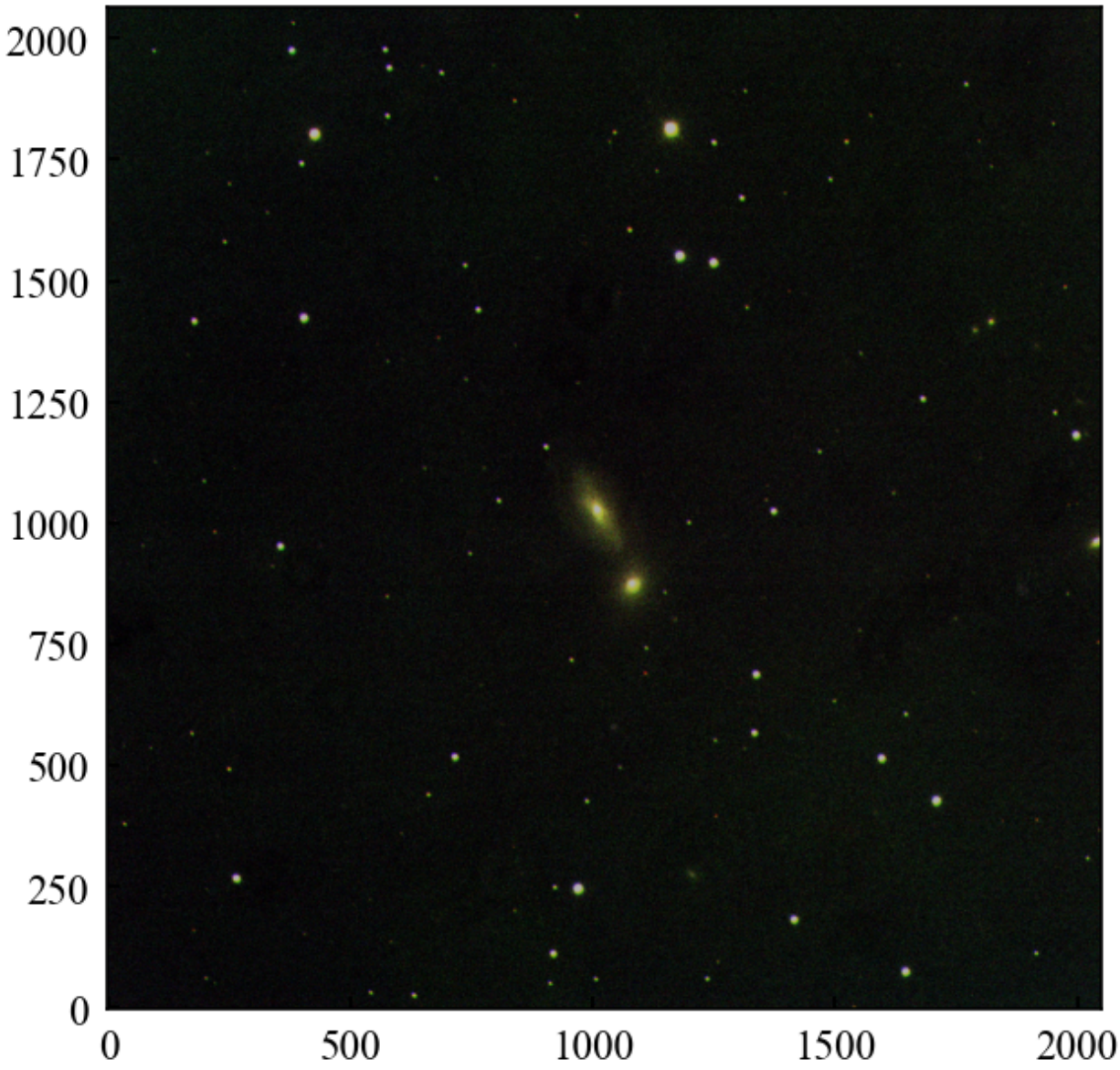


In [67]:
```
iband2.close()
rband2.close()
gband2.close()
```

NGC 3227

In [68]:
```
iband3 = fits.open("/Users/jackcolvin//ngc3227_i-band_120.0s_bin1_250429_051
gband3 = fits.open("/Users/jackcolvin//ngc3227_g-band_120.0s_bin1_250429_051
rband3 = fits.open("/Users/jackcolvin//ngc3227_r-band_120.0s_bin1_250429_051
```

In [69]:
```
r3 = iband3[0].data
g3 = rband3[0].data
b3 = gband3[0].data

#lupton_rgb doesn't support scaling
#so that must be done manually
zscale = ZScaleInterval()
r_scaled = zscale(r3)
g_scaled = zscale(g3)
b_scaled = zscale(b3)



image = make_lupton_rgb(r_scaled, g_scaled, b_scaled, Q = 0, stretch = 1, mi
plt.imshow(image, origin = 'lower')
plt.show()
```

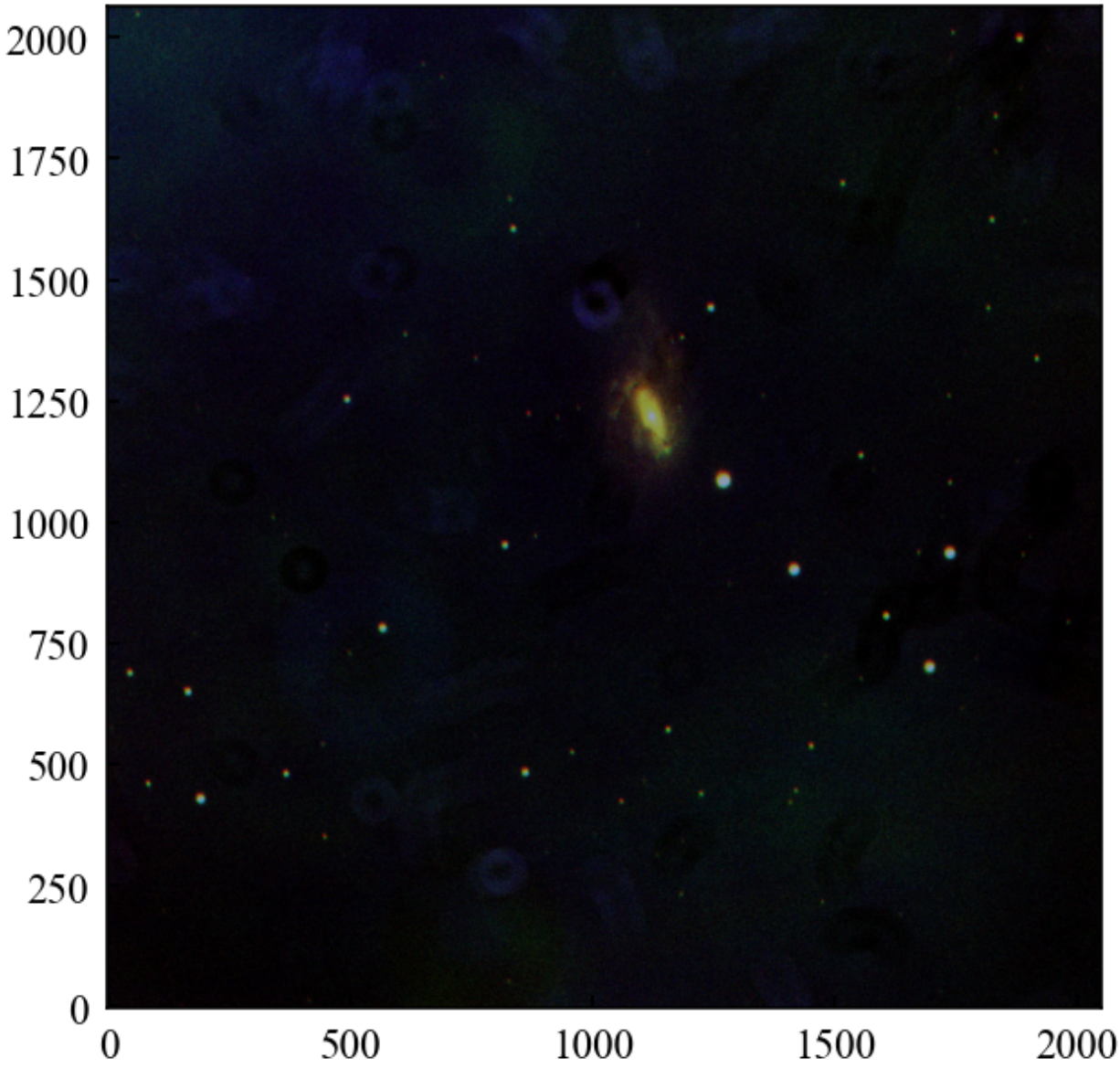


In [70]:
```
iband3.close()
rband3.close()
gband3.close()
```

M66

In [71]:
```
iband4 = fits.open("/Users/jackcolvin//m66_i-band_120.0s_bin1_250430_035359_
gband4 = fits.open("/Users/jackcolvin//m66_g-band_120.0s_bin1_250430_034837_
rband4 = fits.open("/Users/jackcolvin//m66_r-band_120.0s_bin1_250430_035112_
```

In [72]:
```
r4 = iband4[0].data
g4 = rband4[0].data
b4 = gband4[0].data

#lupton_rgb doesn't support scaling
#so that must be done manually
zscale = ZScaleInterval()
r_scaled = zscale(r4)
g_scaled = zscale(g4)
b_scaled = zscale(b4)



image = make_lupton_rgb(.85*r_scaled, .9*g_scaled, b_scaled, Q = 0, stretch
plt.imshow(image, origin = 'lower')
plt.show()
```

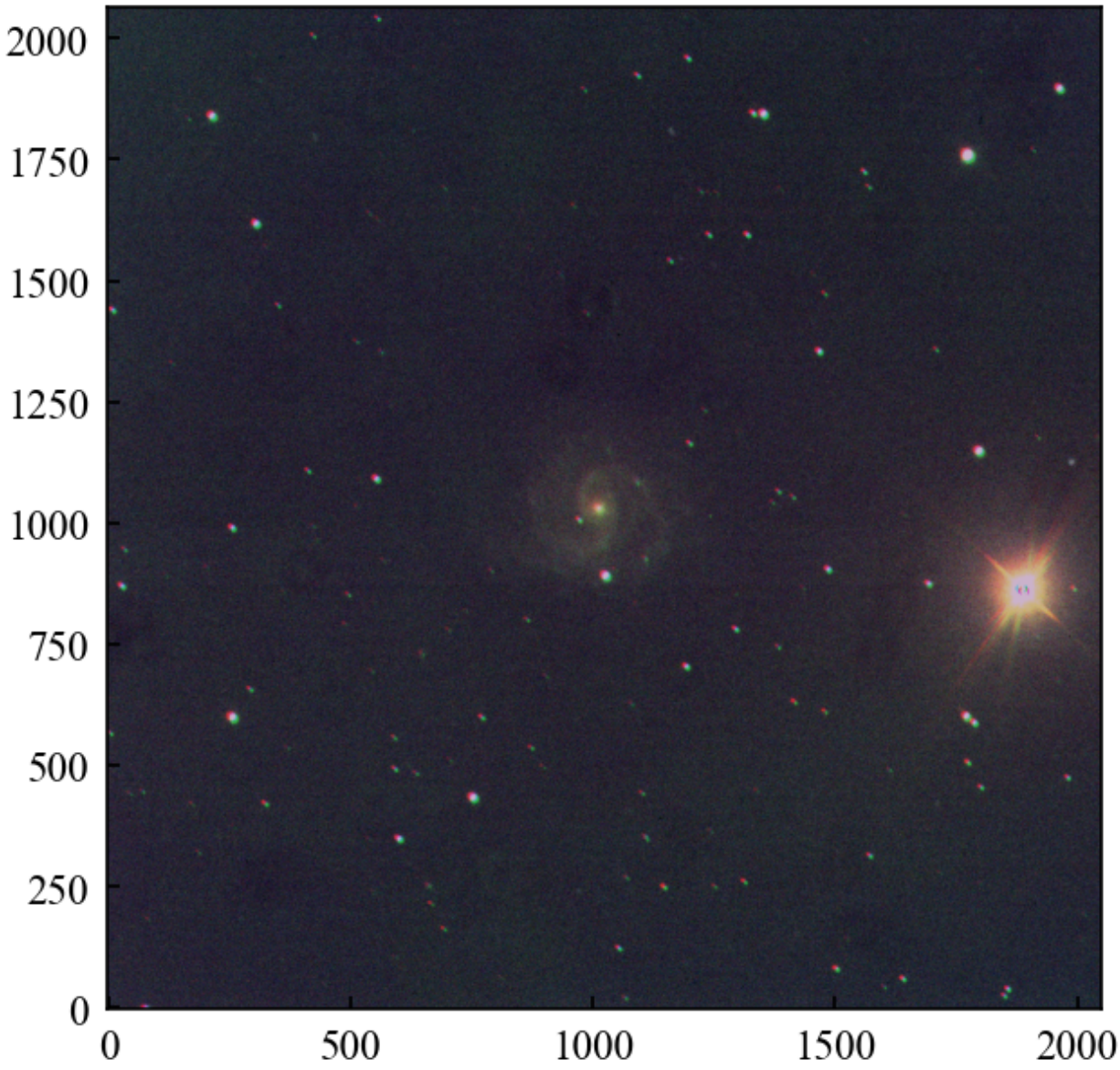


In [73]:
```
iband4.close()
rband4.close()
gband4.close()
```

NGC 3180

In [74]:
```
iband5 = fits.open("/Users/jackcolvin//ngc3180_i-band_120.0s_bin1_250429_044
gband5 = fits.open("/Users/jackcolvin//ngc3180_g-band_120.0s_bin1_250429_044
rband5 = fits.open("/Users/jackcolvin//ngc3180_r-band_120.0s_bin1_250429_044
```

In [75]:
```
r5 = iband5[0].data
g5 = rband5[0].data
b5 = gband5[0].data

#lupton_rgb doesn't support scaling
#so that must be done manually
zscale = ZScaleInterval()
r_scaled = zscale(r5)
g_scaled = zscale(g5)
b_scaled = zscale(b5)



image = make_lupton_rgb(r_scaled, .9*g_scaled, b_scaled, Q = 0, stretch = 1,
plt.imshow(image, origin = 'lower')
plt.show()
```

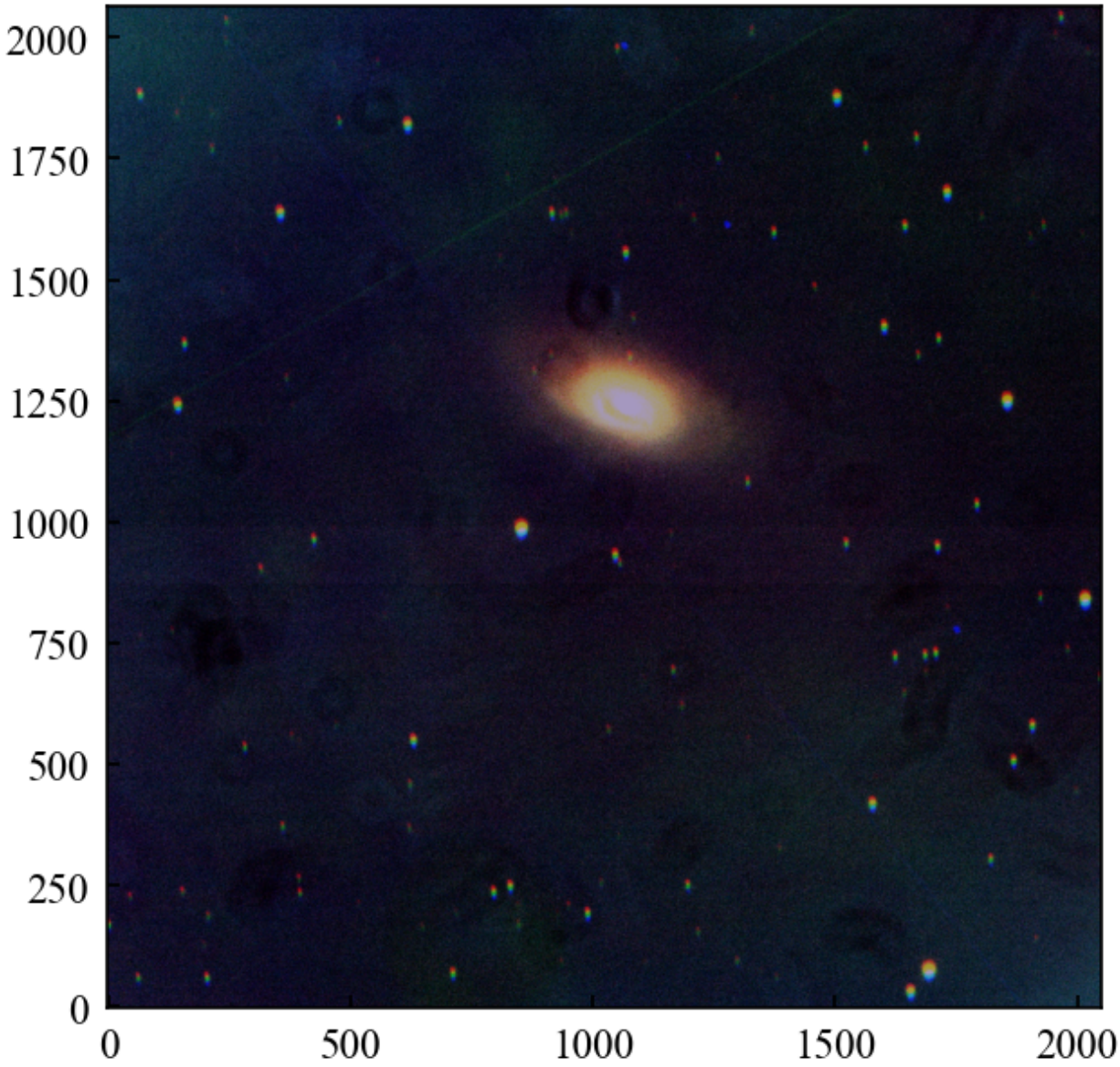


In [76]:
```
iband5.close()
rband5.close()
gband5.close()
```

M64

In [77]:
```
iband6 = fits.open("/Users/jackcolvin//m64_i-band_500.0s_bin1_250418_041143_
gband6 = fits.open("/Users/jackcolvin//m64_g-band_500.0s_bin1_250418_035254_
rband6 = fits.open("/Users/jackcolvin//m64_r-band_500.0s_bin1_250418_040246_
```

In [80]:
```
r6 = iband6[0].data
g6 = rband6[0].data
b6 = gband6[0].data

#lupton_rgb doesn't support scaling
#so that must be done manually
zscale = ZScaleInterval()
r_scaled = zscale(r6)
g_scaled = zscale(g6)
b_scaled = zscale(b6)



image = make_lupton_rgb(.9*r_scaled, .85*g_scaled, b_scaled, Q = 0, stretch
plt.imshow(image, origin = 'lower')
plt.show()
```

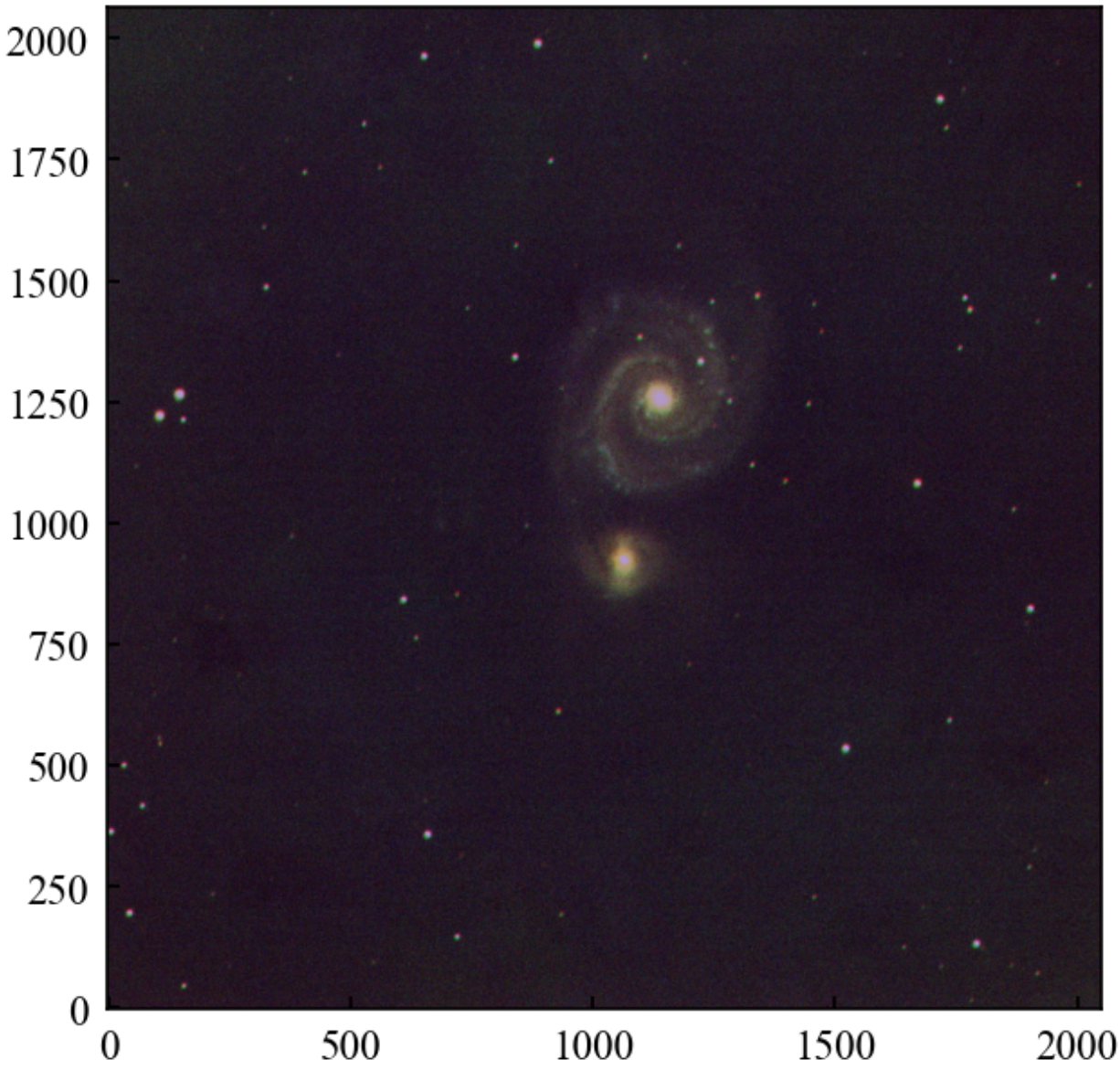


In [81]:
```
iband6.close()
rband6.close()
gband6.close()
```

M51

In [82]:
```
iband7 = fits.open("/Users/jackcolvin//m51_i-band_60.0s_bin1_250422_044715_k
gband7 = fits.open("/Users/jackcolvin//m51_g-band_60.0s_bin1_250422_044459_k
rband7 = fits.open("/Users/jackcolvin/m51_r-band_60.0s_bin1_250422_044326_kl
```

In [83]:
```
r7 = iband7[0].data
g7 = rband7[0].data
b7 = gband7[0].data

#lupton_rgb doesn't support scaling
#so that must be done manually
zscale = ZScaleInterval()
r_scaled = zscale(r7)
g_scaled = zscale(g7)
b_scaled = zscale(b7)



image = make_lupton_rgb(.9*r_scaled, .8*g_scaled, b_scaled, Q = 0, stretch =
plt.imshow(image, origin = 'lower')
plt.show()
```

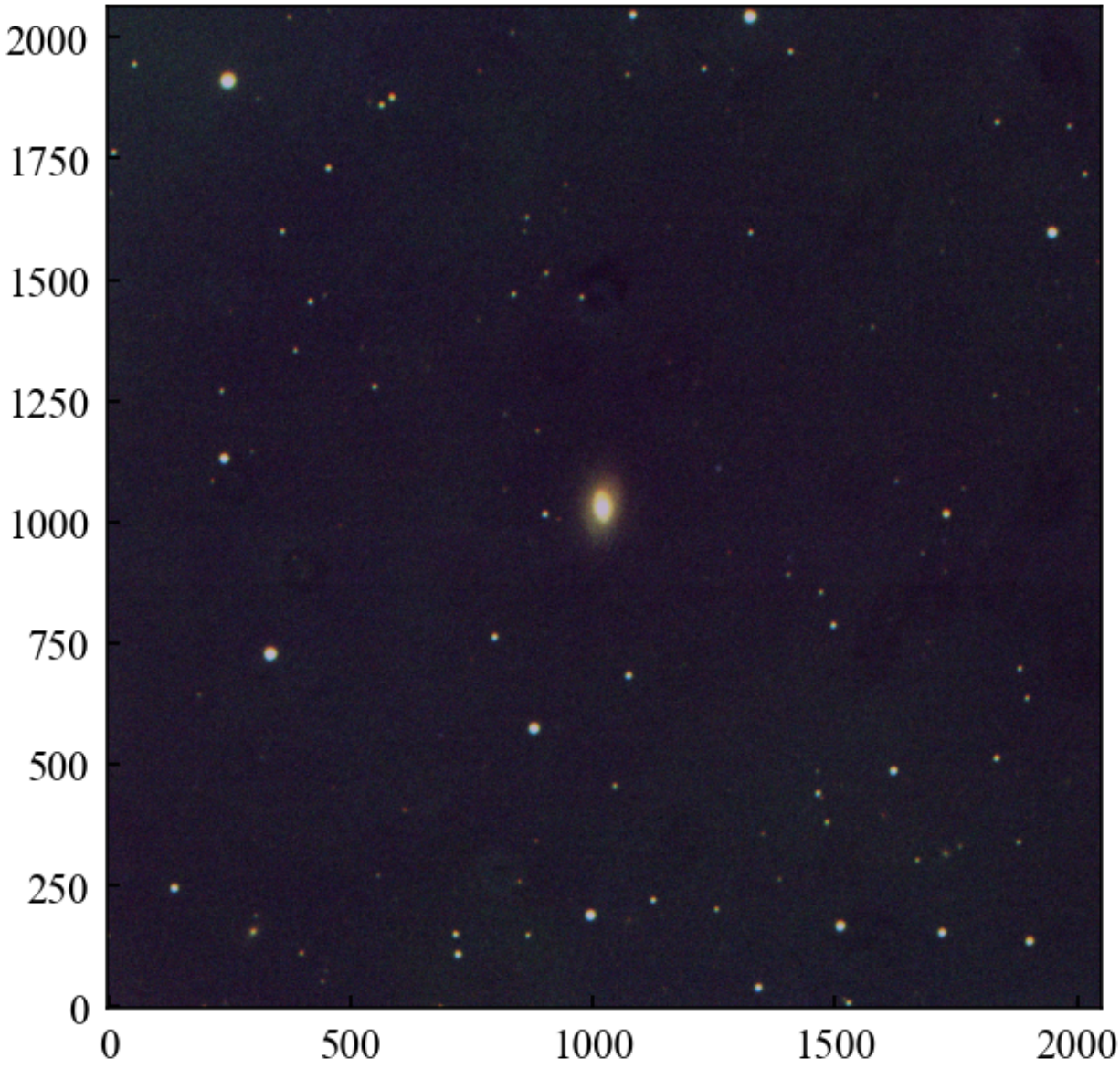


In [84]:
```
iband7.close()
rband7.close()
gband7.close()
```

NGC 3941

In [85]:
```
iband8 = fits.open("/Users/jackcolvin//ngc3941_i-band_120.0s_bin1_250501_043
gband8 = fits.open("/Users/jackcolvin//ngc3941_g-band_120.0s_bin1_250501_043
rband8 = fits.open("/Users/jackcolvin//ngc3941_r-band_120.0s_bin1_250501_043
```

In [86]:
```
r8 = iband8[0].data
g8 = rband8[0].data
b8 = gband8[0].data

#lupton_rgb doesn't support scaling
#so that must be done manually
zscale = ZScaleInterval()
r_scaled = zscale(r8)
g_scaled = zscale(g8)
b_scaled = zscale(b8)



image = make_lupton_rgb(.95*r_scaled, .9*g_scaled, b_scaled, Q = 0, stretch
plt.imshow(image, origin = 'lower')
plt.show()
```

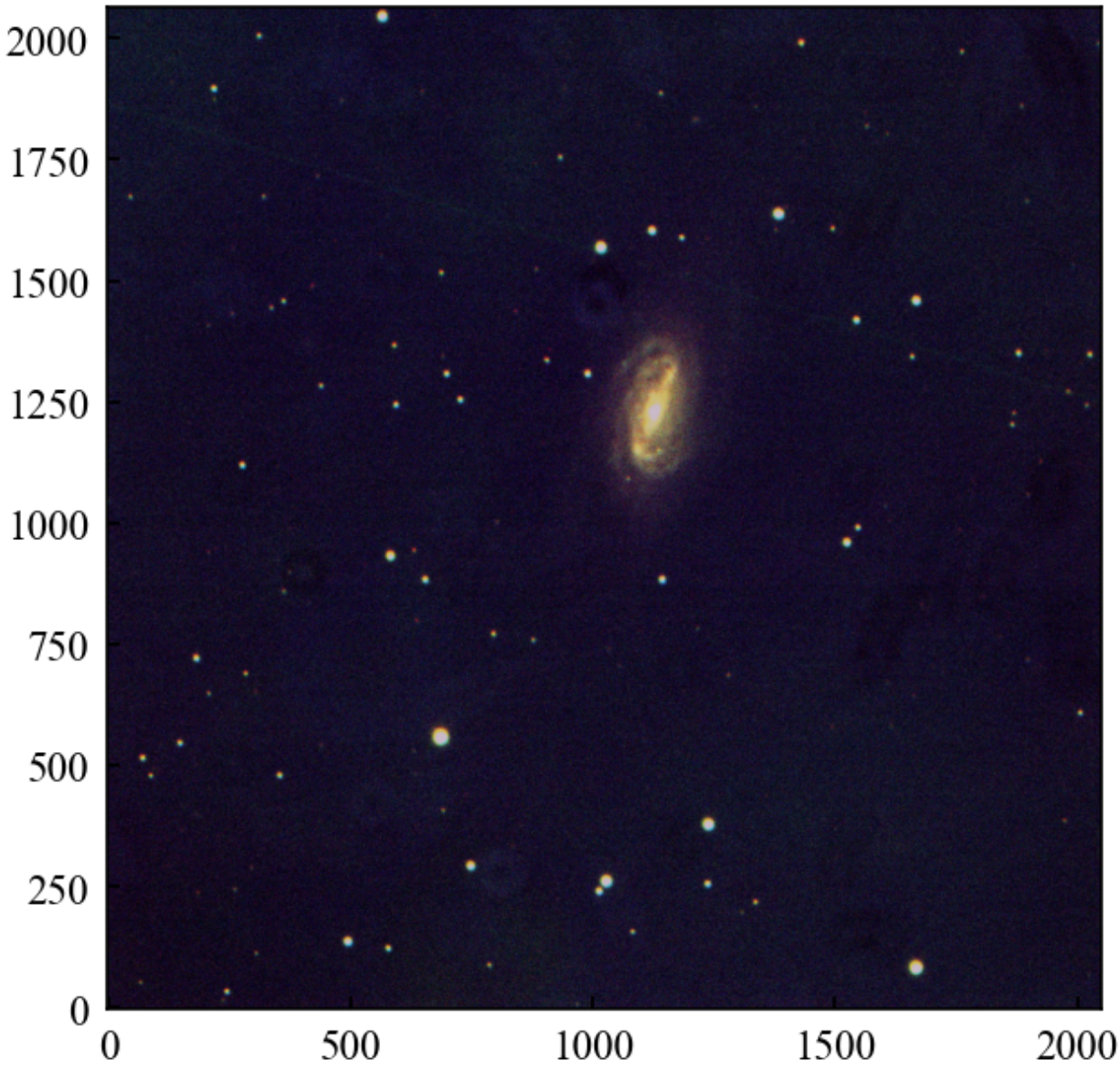


In [87]:
```
iband8.close()
rband8.close()
gband8.close()
```

NGC 2905

In [88]:
```
iband9 = fits.open("/Users/jackcolvin//ngc2905_i-band_120.0s_bin1_250501_041
gband9 = fits.open("/Users/jackcolvin//ngc2905_g-band_120.0s_bin1_250501_040
rband9 = fits.open("/Users/jackcolvin//ngc2905_r-band_120.0s_bin1_250501_041
```

In [89]:
```
r9 = iband9[0].data
g9 = rband9[0].data
b9 = gband9[0].data

#lupton_rgb doesn't support scaling
#so that must be done manually
zscale = ZScaleInterval()
r_scaled = zscale(r9)
g_scaled = zscale(g9)
b_scaled = zscale(b9)



image = make_lupton_rgb(.9*r_scaled, .85*g_scaled, .9*b_scaled, Q = 1, stret
plt.imshow(image, origin = 'lower')
plt.show()
```



In [90]:
```
iband9.close()
rband9.close()
gband9.close()
```

Part One Conclusion:

It was definitely much smoother making the RGB images this time around, given that we already had experience working with scaling and what effect changing specific parameters has on our images, althought some weren't quite as vibrant as I would have liked. I was also interested in the background colors of the base images. All 4 of our observations, which were taken on the same day at similar times, had very different colors in the background, with one being reddish, another being bluesh, another being greenish, and our last one being black. We hypothesized that potentially the weather conditions changing throughout our observations could have possibly accounted for this, but we were still a little bit confused as to how the background colors of our images changed so rapidly throughout the ~1 hour we were observing. Additionally, I had some problems with scaling bcause of there being extremely bright objects in the background, particularly for ngc 3180, which drowned out the light from the objects themselves.

Part Two: Analyzing Our Images!

In our analysis, we were trying to find a consistent method to measure angular extent of our galaxies which we could then use to find distance of our galaxies to compare to our redshifts and from there discern a measure of the hubble constant. I decided to use ds9 for this analysis, specifically because of the built in contour feature, which draws contours that naturally divide the image into regions from brightest to darkest, which I then used to discern where the distincitons between the background and the object was. I then used the vector feature to measure the size of the contour. There were several considerations I kept in mind throughout this process to keep my analysis as consistent as possible between different galaxies:

- I made sure to use the same number for contours and smoothness (10 and 10) consistent throughout all the images to keep the specific contour I was using constant across all of the images
- I used the i-band filter, as based upon visiual analysis, it gave the most generous approximation of angular extent, as the light in the i-band seemed to spread out the most (though all bands were very similar and any difference here is likely negligible)
- I cropped the image to include just the galaxy I was looking at, as some of the stars in the image were brighter than the galaxy itself, which was biasing the contours for a couple of the images, as they were focusing on the brighter object and producing

less detail on the object I was considering (Ex. NGC 3180).

- Some of the galaxies were elliptical in shape, potentially indicating they were being viewed at a sideways angle, and thus I always used the major axis of the ellipse so as to capture the longer side of the galaxy as a better approximation of the true angular extent of the galaxy.
- I used zscale with squared rather than linear scaling in order to increase the brightness of the objects relative to the background, as sqaured scaling did a very good job of dimming the background while keeping the angular extent of the galaxy the same, which made a much clearer definition between object and backround (For the first couple of galaxies, I included an image of the galaxy with linear vs sqaured scaling to show how useful this scaling was). This was the closest I could get to the "subtracting the median" strategy used by some of my group mates in python, as I could not figure out how to perfectly replicate that strategy in ds9.
- I always used the same contour for all of the galaxies, namely I used the widest contour before the background to capture all of the light from the galaxy up untill it was indistinguishable from the background (and visually, this strategy matched up very well to qualitative analysis of where the galaxy crossed into the background).
- Overall, me and my groupmates in ds9 vs python tried to use methods that were as similar as possible in order to get results that were comparable to each other.
- There could be inherent systematic error here, as I used a uniform place among all galaxies to distinguish between background and my image, however that uniform cut off could be an overestimation or an underestimation of the true angular size of my galaxies, which in turn would create a systematic shift in the values for all of my galaxies.

After doing this analysis, we used the formula provided by the lab instructions: $angular size = \frac{22kpc}{distance}$ to convert our calculated angular sizes to distance, assuming all galaxies are the same size and thus can be used as standard rulers, which we compared to the known redshifts of all of our galaxies. Finally, we graphed our galaxies and used the slope of a linear best fit line to calculate the hubble constant as measured from out galaxies.

Note: I also tried a similar strategy, but implimenting binning, in which I recieved different yet interesting results, which I have included and discussed in section 4.1 of this lab report.

Angular Size Measurements!

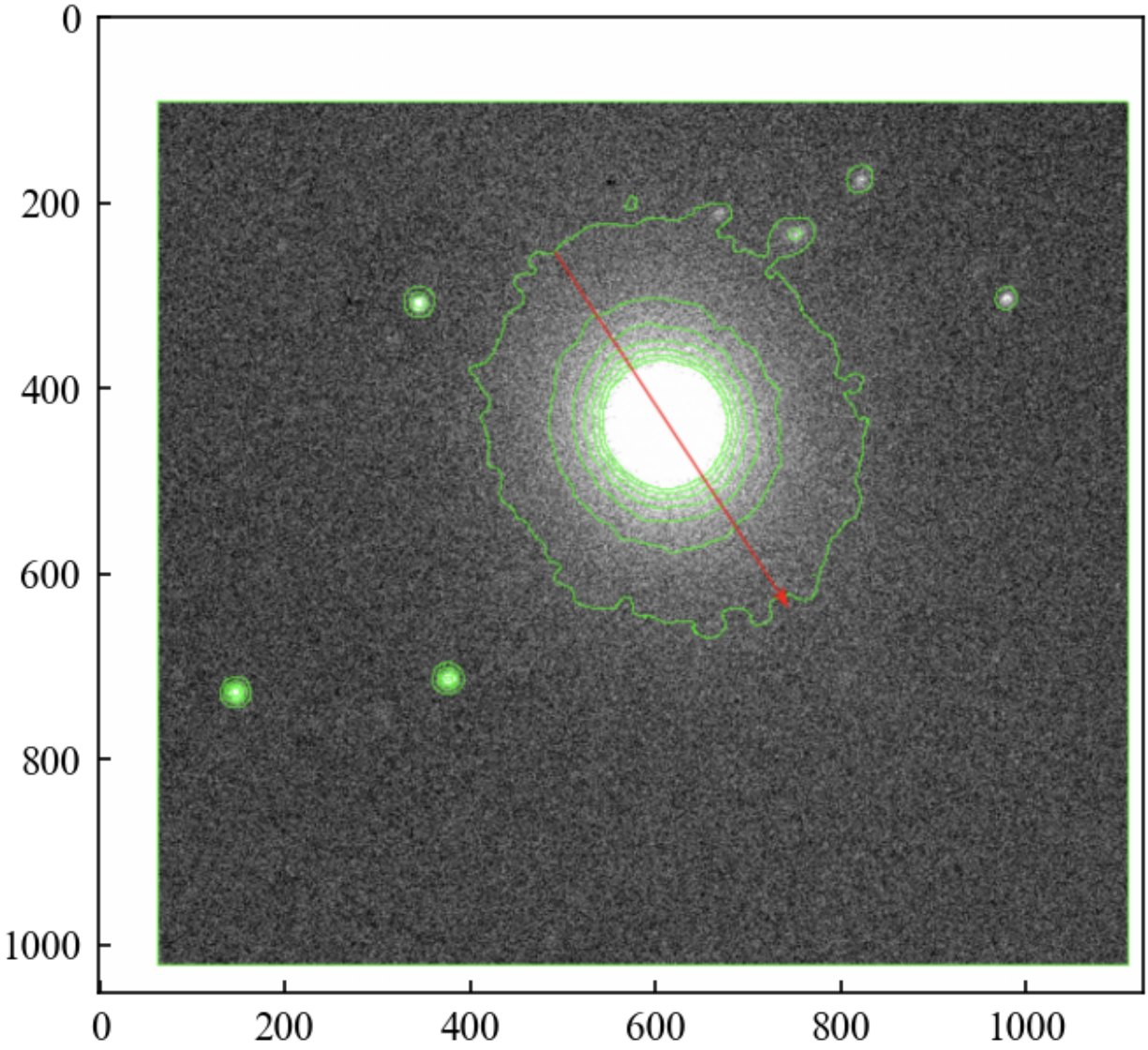
Below is a table of all of my results for angular size. I have also included images from ds9 that show exactly how i got to each of these values using my method.

Name	Ra	Dec	Morphology	Angular Size
m87	12h30m49s	12d23m28s	lenticular	.0568
m96	10h46m46s	12d05m02s	spiral	.0556
ngc3516	11h06m48s	72d34m07s	elliptical	.0253
ngc3227	10h23m31s	19d51m54s	elliptical	.0484
m51	13h27m50s	47d13m50s	spiral	.0874
m64	12h56m44s	21d40m59s	elliptical	.136
ngc3180	10h18m10s	41d26m42s	spiral	.0670
m66	11h20m15s	12d59m29s	elliptical	.0652
ngc2095	05h42m51s	-67d19m18s	spiral	.0753
ngc3941	11h52m55s	36d59m11s	elliptical	.0506

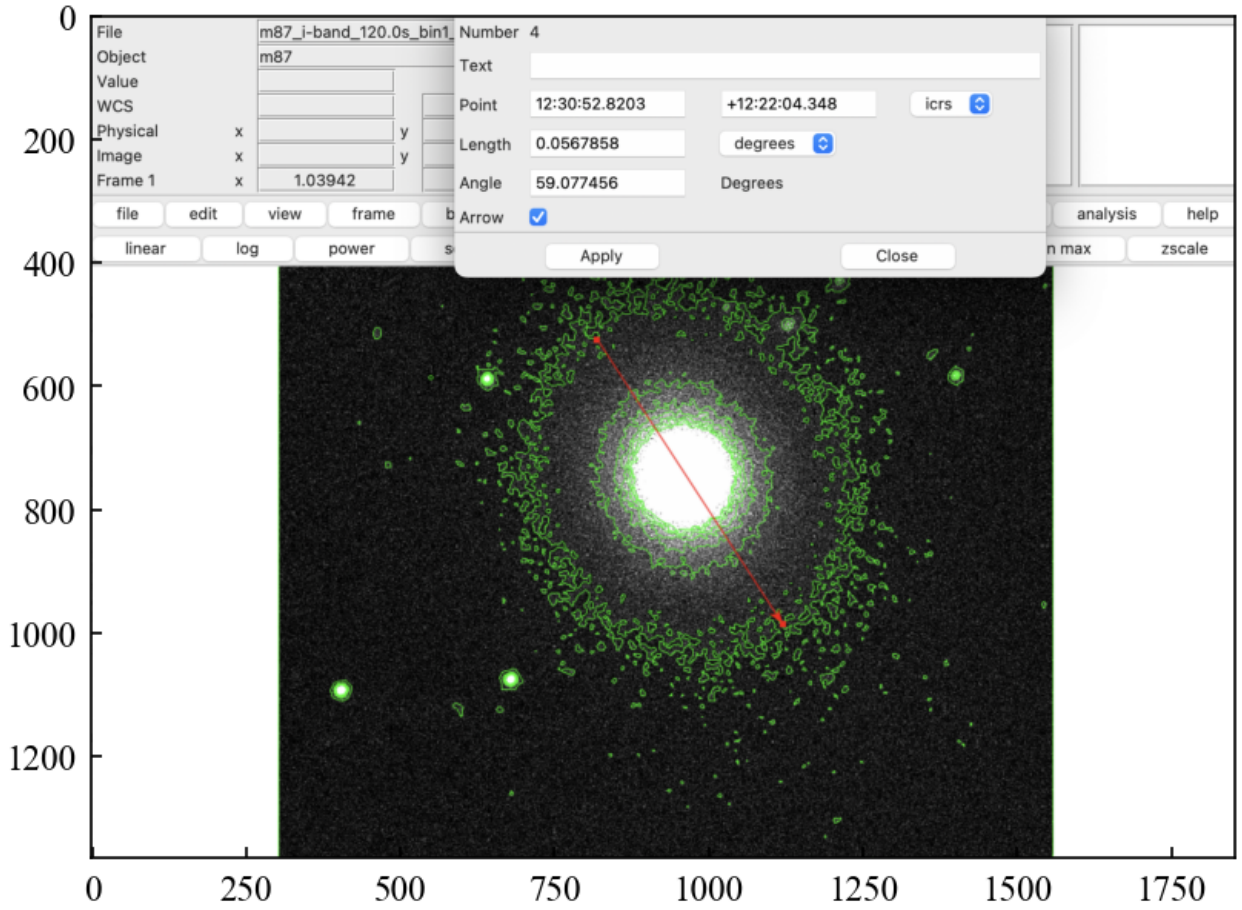
M87:

Calculated angular size = .0568, Redshift = .004283

In [91]:
```
M87_img = Image.open("/Users/jackcolvin//Screenshot 2025-05-01 at 5.34.24 PM
M87_img1 = Image.open("/Users/jackcolvin//Screenshot 2025-05-01 at 5.18.10 P

plt.imshow(M87_img)
plt.show()
plt.imshow(M87_img1)
```



Out[91]: <matplotlib.image.AxesImage at 0x16e471010>



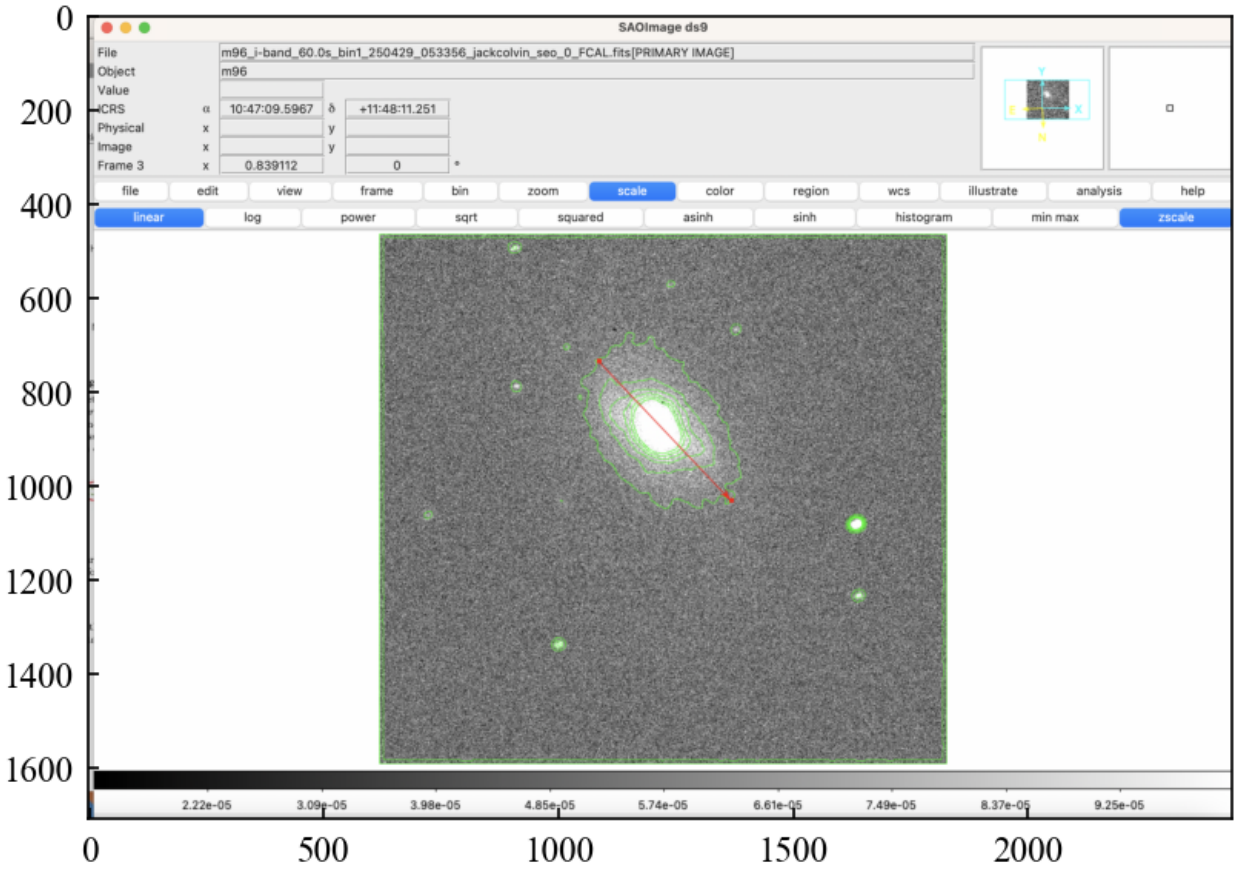
In [92]:
```
M87_img.close()
M87_img1.close()
```

As we can see, both images have very similar angular extents, but the second, which uses sqaure instead of linear scaling along withr zscale, has a much clearer boundary between background and object. This is why I decided to go with this scaling factor, as the rest either made the object smaller, or it was very difficult to discern where the cut-off was between background and object.

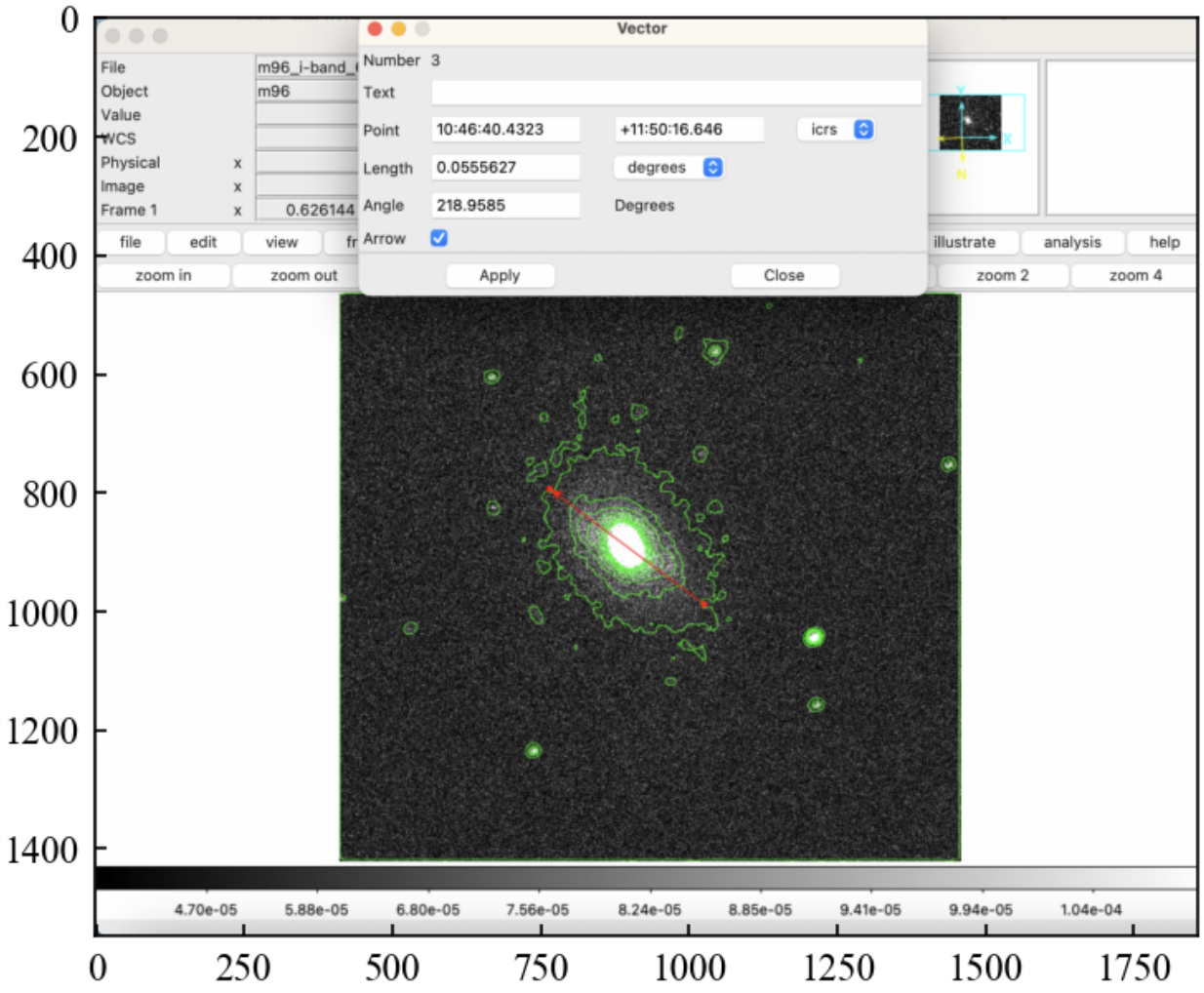
M96:

Calculated angular size = .0556, Redshift = .002967

In [93]:
```
M96_img = Image.open("/Users/jackcolvin//Screenshot 2025-05-05 at 4.39.31 PM
M96_img1 = Image.open("/Users/jackcolvin//Screenshot 2025-05-01 at 5.36.04 P

plt.imshow(M96_img1)
plt.show()
plt.imshow(M96_img)
```



Out[93]: <matplotlib.image.AxesImage at 0x16e515e50>



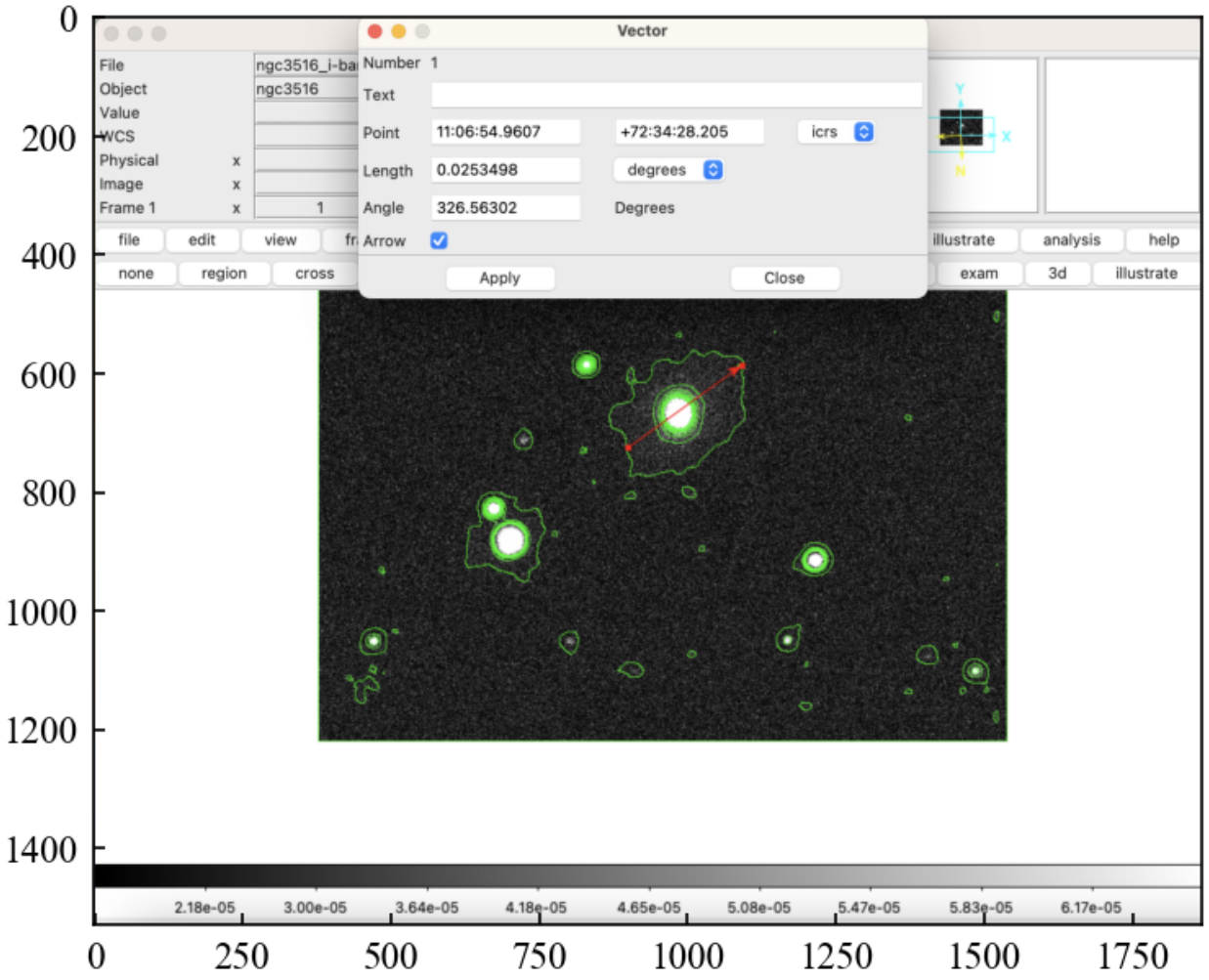
In [94]:
```
M96_img.close()
M96_img1.close()
```

NGC 3516

Calculated angular size = .0212, Redshift = .008836

In [95]:
```
NGC3516_img = Image.open("/Users/jackcolvin//Screenshot 2025-05-05 at 3.37.0

plt.imshow(NGC3516_img)
```

Out[95]:
```
<matplotlib.image.AxesImage at 0x16e450310>
```



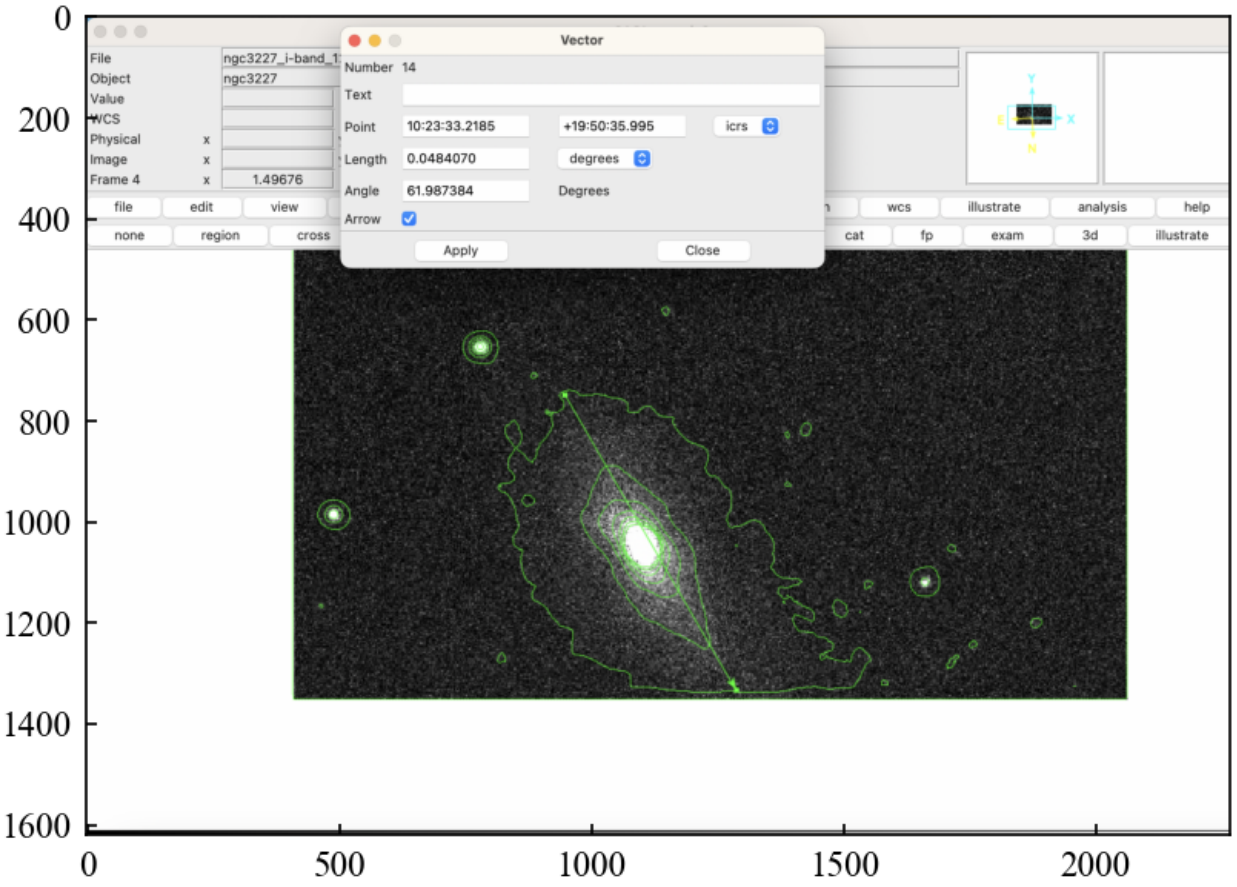
In [96]:
```
NGC3516_img.close()
```

NGC 3227

Calculated angular size = .048, Redshift = .003756

In [97]:
```
NGC3227_img = Image.open("/Users/jackcolvin//Screenshot 2025-05-01 at 5.45.3

plt.imshow(NGC3227_img)
```

Out[97]:
```
<matplotlib.image.AxesImage at 0x16e532c50>
```



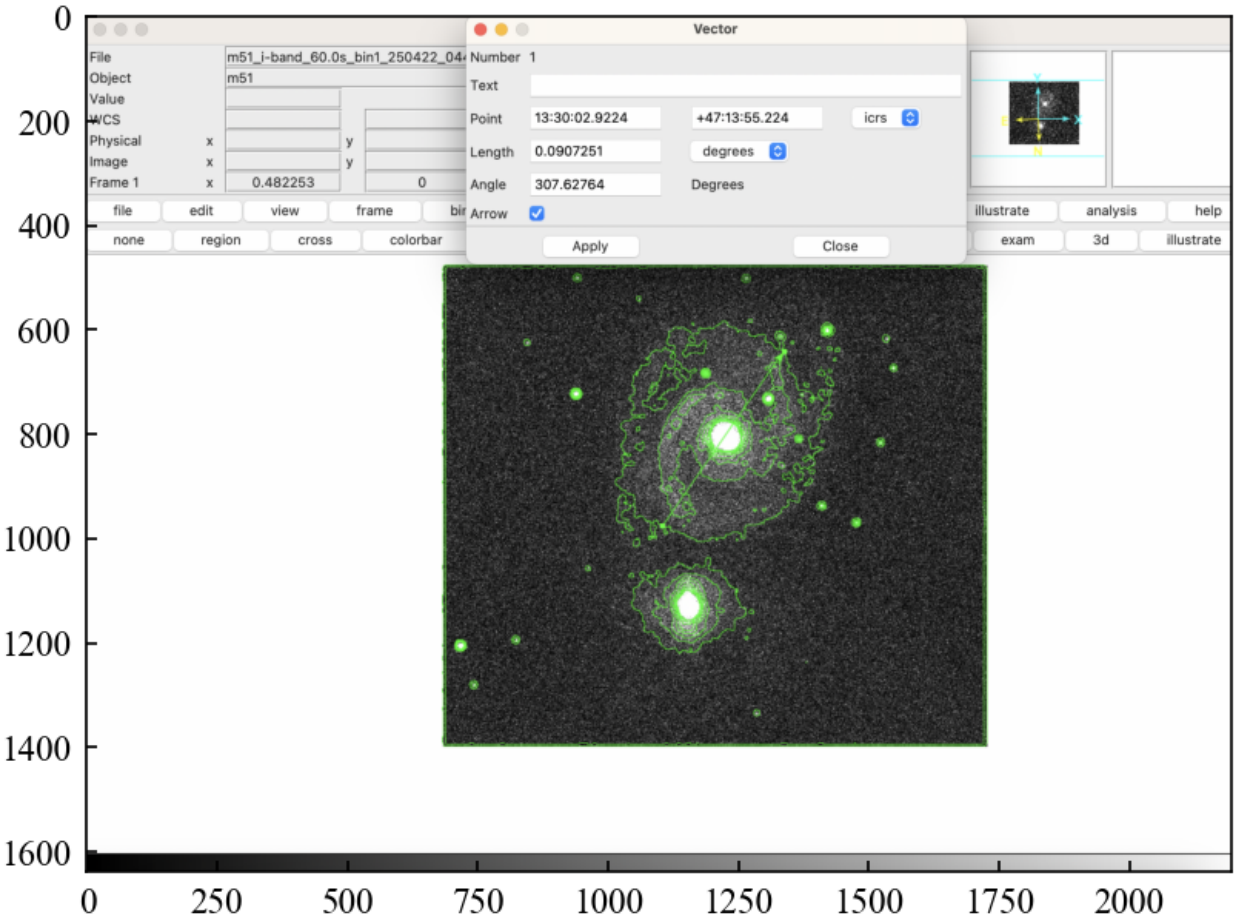
In [98]:
```
NGC3227_img.close()
```

M51

Calculated angular size = .0874, Redshift = .001543

In [99]:
```
M51_img = Image.open("/Users/jackcolvin//Screenshot 2025-05-05 at 5.01.46 PM

plt.imshow(M51_img)
```

Out[99]:
```
<matplotlib.image.AxesImage at 0x16e3dfbd0>
```



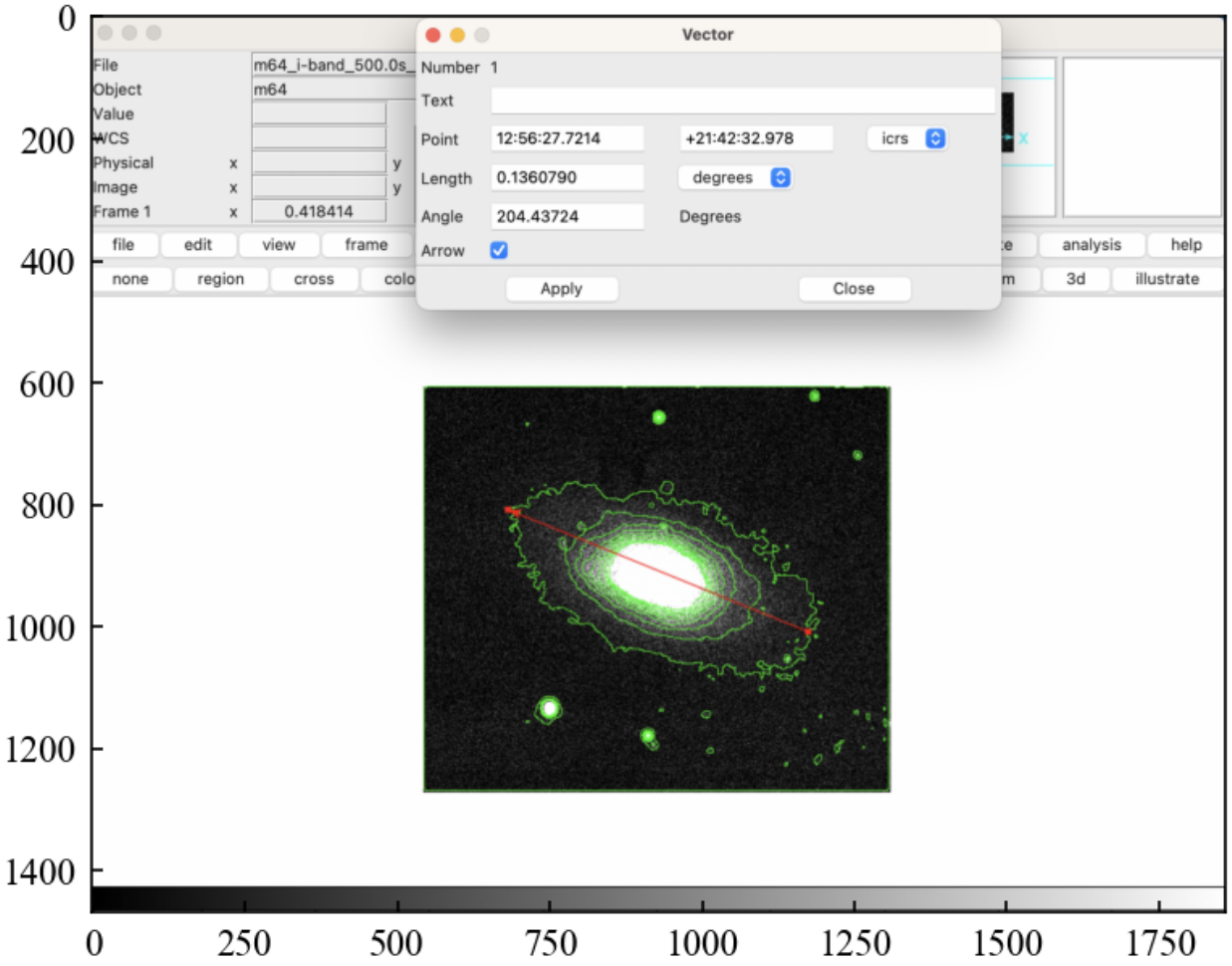
In [100…
```
M51_img.close()
```

M64

Calculated angular size = .136 Redshift = .001361

In [101…
```
M64_img = Image.open("/Users/jackcolvin/Screenshot 2025-05-04 at 3.12.43 PM.

plt.imshow(M64_img)
```

Out[101]: <matplotlib.image.AxesImage at 0x16e1e0fd0>

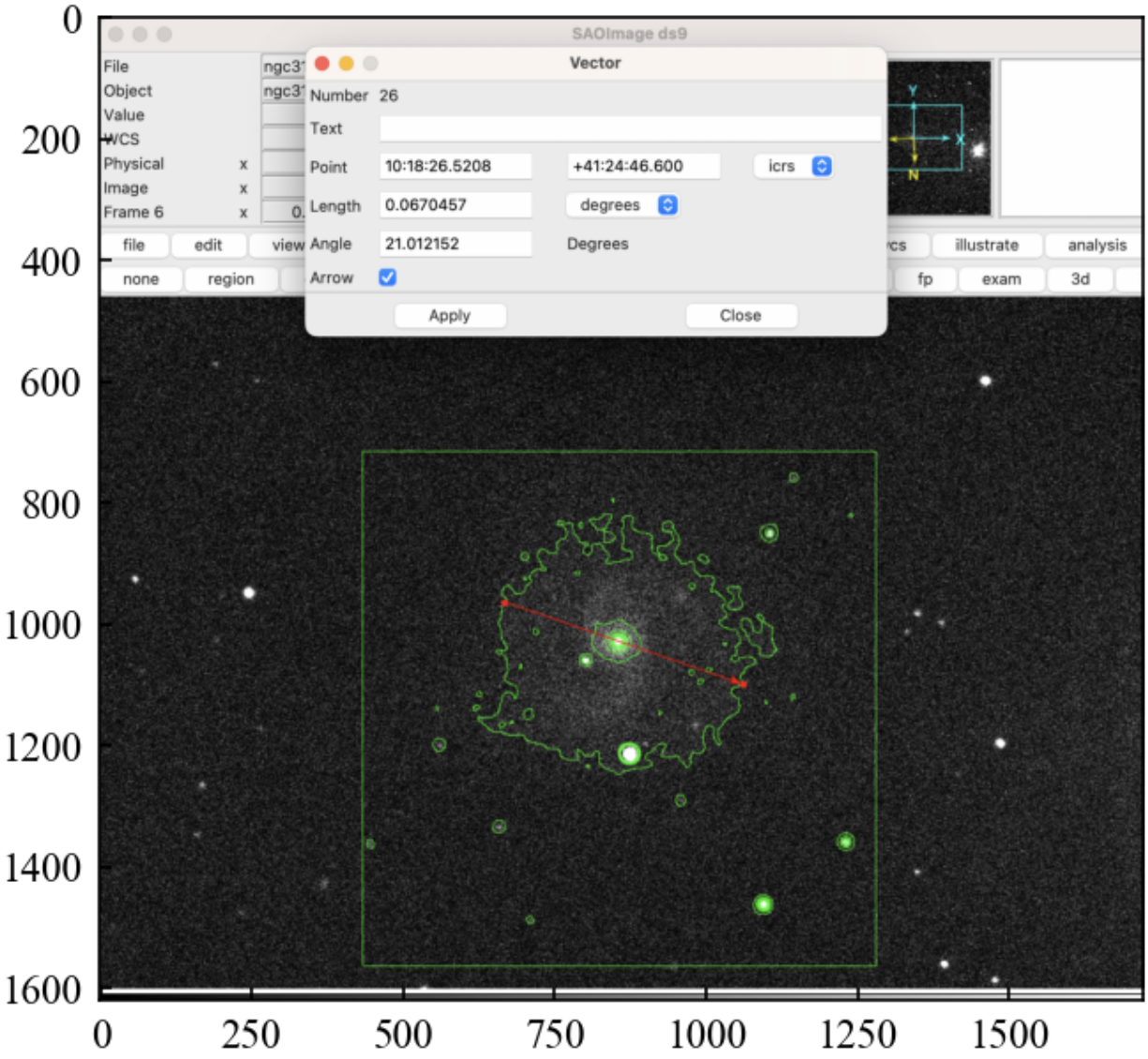


NGC 3180

Calculated angular size = .0670 Redshift = .001975

In [102…
```
NGC3180_img = Image.open("/Users/jackcolvin//NGC 3180.png")

plt.imshow(NGC3180_img)
```

Out[102]:
```
<matplotlib.image.AxesImage at 0x16e0fe7d0>
```



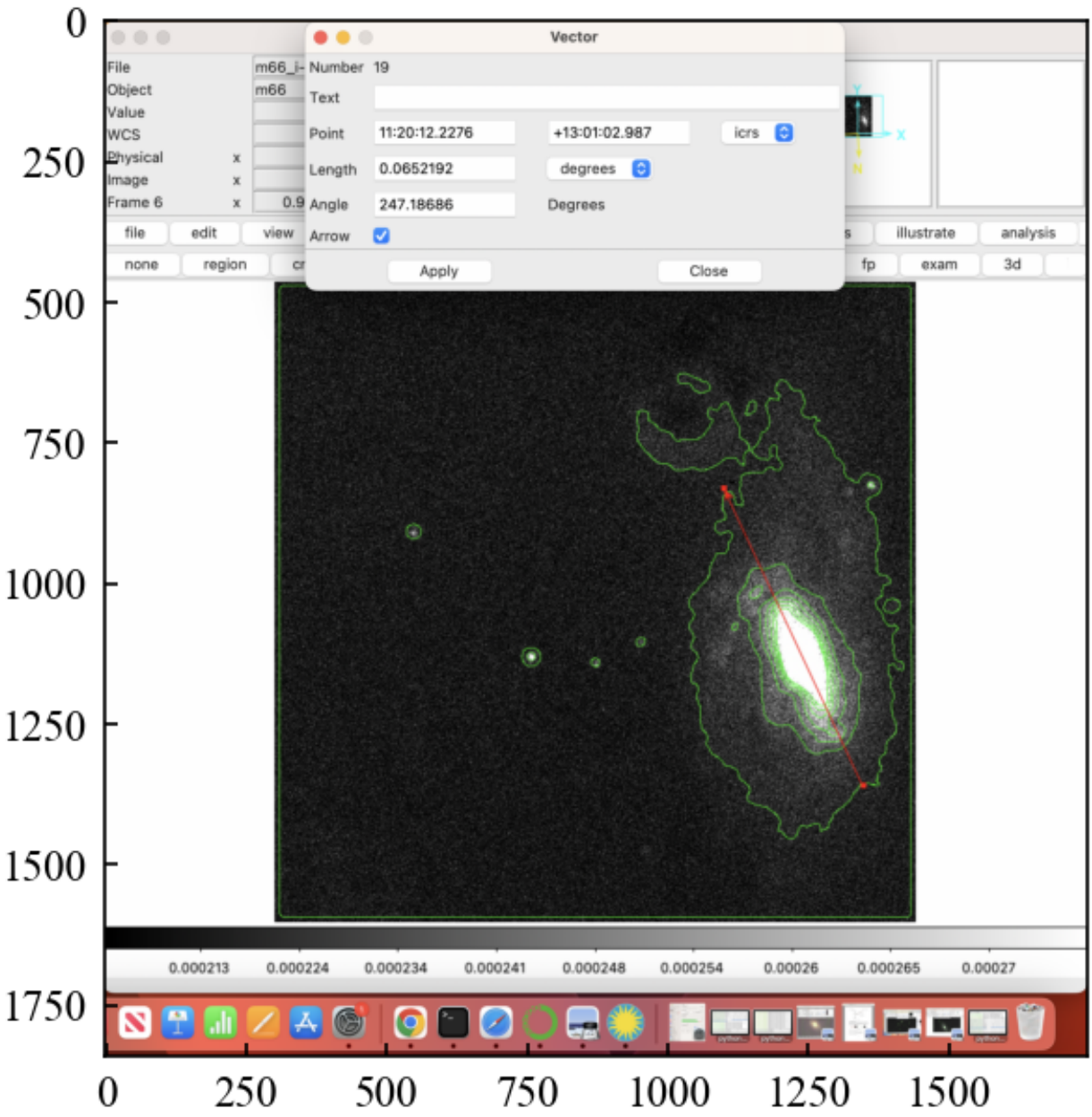
In [103…
```
NGC3180_img.close()
```

M66

Calculated angular size = .0652 Redshift = .002425

In [104…
```
M66_img = Image.open("/Users/jackcolvin//M66.png")

plt.imshow(M66_img)
```

Out[104]:
```
<matplotlib.image.AxesImage at 0x16cfe61d0>
```



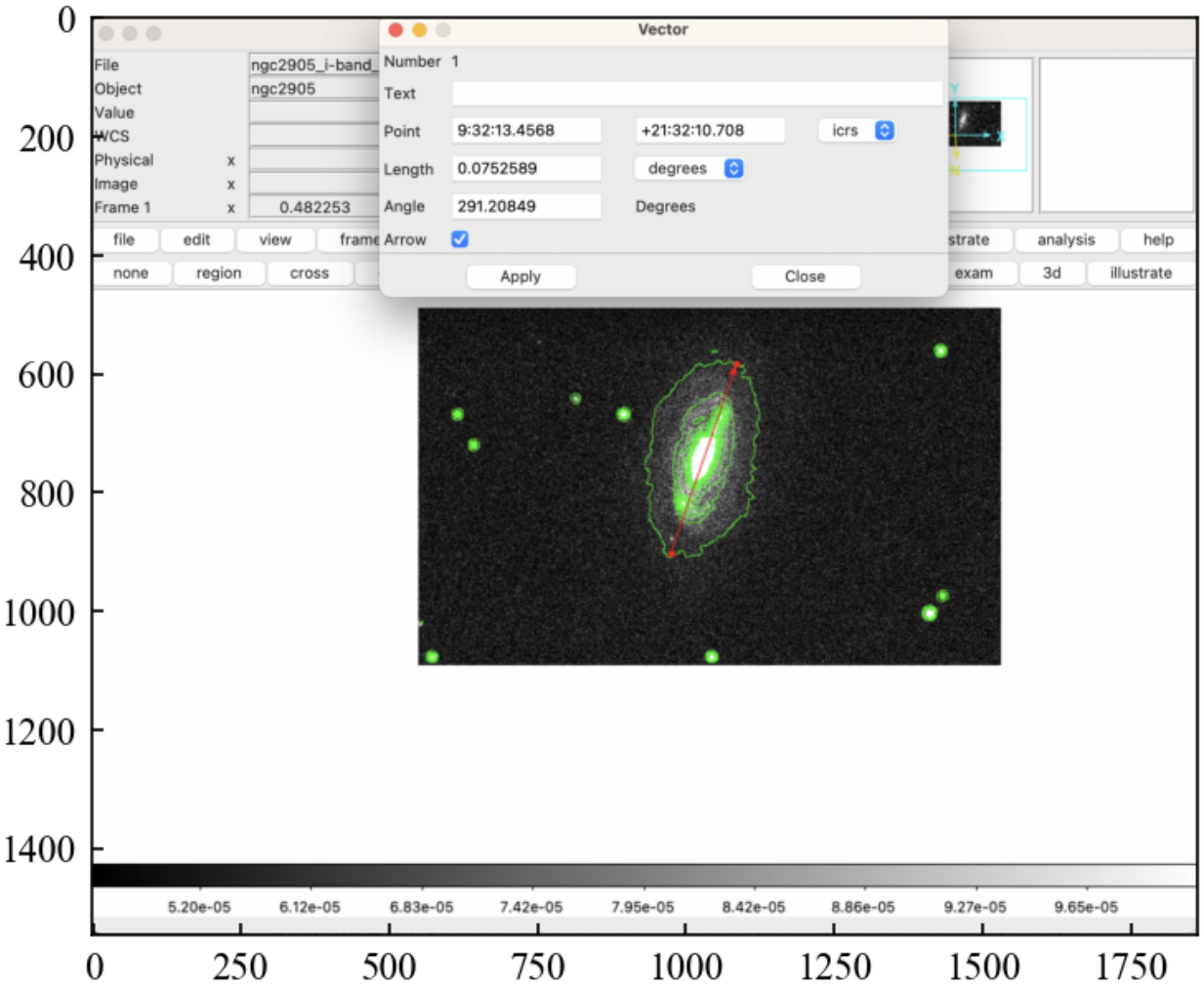
In [105…
```
M66_img.close()
```

NGC2095

Calculated angular size = .0753 Redshift = .001855

In [106…
```
NGC2095_img = Image.open("/Users/jackcolvin//Screenshot 2025-05-04 at 8.12.2

plt.imshow(NGC2095_img)
```

Out[106]:
```
<matplotlib.image.AxesImage at 0x16aeb8990>
```



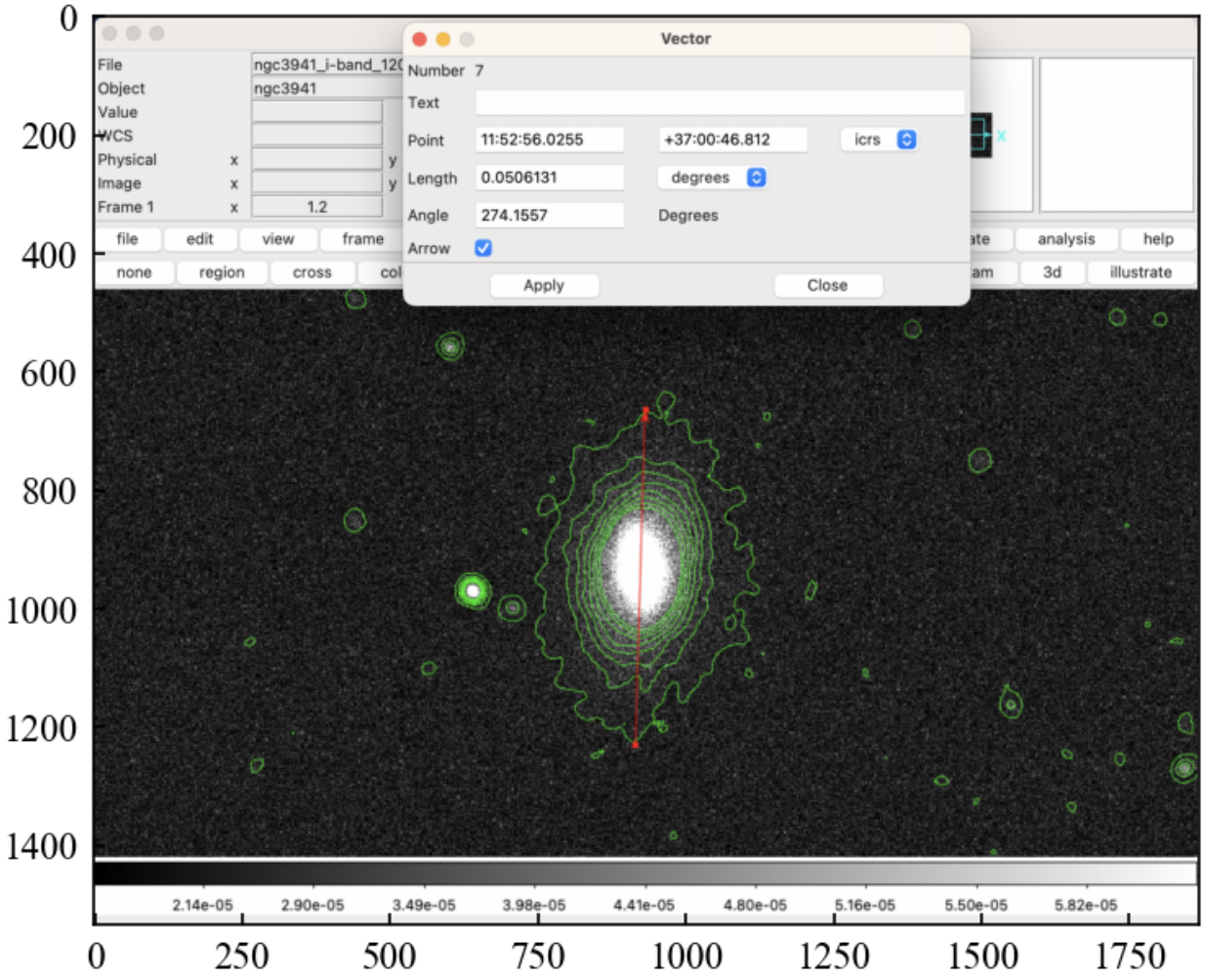
In [107…
```
NGC2095_img.close()
```

NGC 3941

Calculated angular size = .0506 Redshift = .003095

In [108…
```
NGC3941_img = Image.open("/Users/jackcolvin//Screenshot 2025-05-04 at 8.01.0

plt.imshow(NGC3941_img)
```

Out[108]:
```
<matplotlib.image.AxesImage at 0x169dd2c50>
```



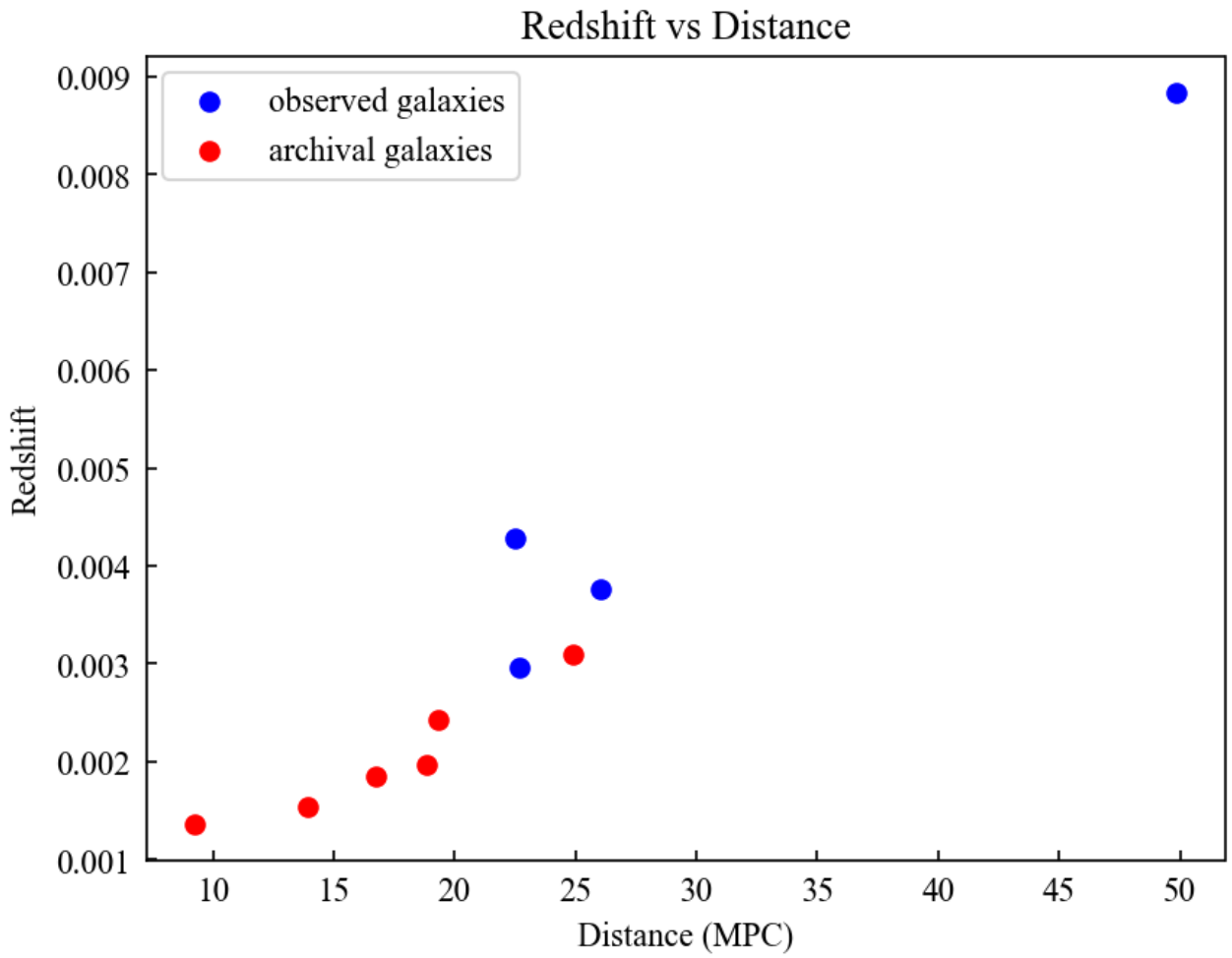
In [109…
```
NGC3941_img.close()
```

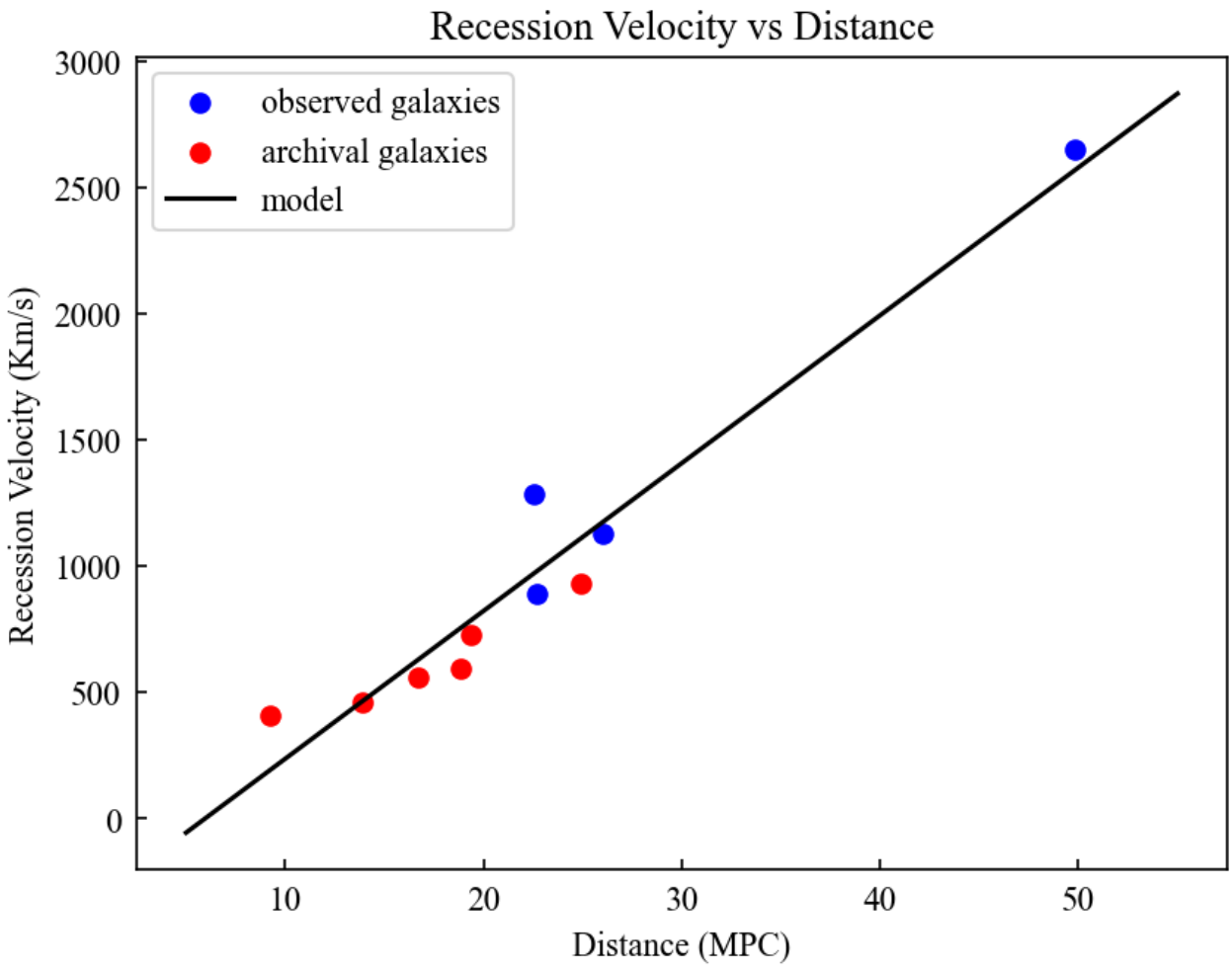
Using polyfit to find our Hubble Constant

The next step is to use our analysis of angular size to calculate the hubble constant. First, we converted our measuremens of degrees into radians, and used the formula described in the introduction ot this section to get an approximtaion of distance. We then turned those distances into megaparsecs, to be consistent with the units of the hubble constant, and plotted our velocity values, calculated as cz, against those distances. finally, we used np.ploy fit to fit a linear model to our data, and calculated its slope as a measure of the hubble constant.

In [110…

```
degrees = np.array([.0560, .0556, .0253, .0484, .0907, .136, .0670, .0652, .

radians = degrees * np.pi/180

distance = 22000/radians

distance = distance * 1e-6

redshift = np.array([.004283, .002967, .008836, .003756, .001543, .001361, .

velocity = 300000 * redshift

plt.scatter(distance[0:4], redshift[0:4], color = 'blue')
plt.scatter(distance[4:10], redshift[4:10], color = 'red')
plt.legend(["observed galaxies", "archival galaxies"])
plt.xlabel('Distance (MPC)')
plt.ylabel("Redshift")
plt.title("Redshift vs Distance")
plt.show()
plt.scatter(distance[0:4], velocity[0:4], color = 'blue')
plt.scatter(distance[4:10], velocity[4:10], color = 'red')
plt.xlabel('Distance (MPC)')
plt.ylabel("Recession Velocity (Km/s)")
plt.title("Recession Velocity vs Distance")

a, b = np.polyfit(distance, velocity, 1)

x = np.linspace(5, 55, 50)
plt.plot(x, a*x+b, color = 'black')
plt.legend(["observed galaxies", "archival galaxies", "model"])
plt.show()
print(f'Our calculated hubble constant is {a:.1f}')
```


Redshift vs Distance
observed galaxies
archival galaxies
Redshift
0.009
0.008
0.007
0.006
0.005
0.004
0.003
0.002
0.001
10
15
20
25
30
35
40
45
50
Distance (MPC)



Our calculated hubble constant is 58.6

Part two Conclusions:

- We certainly see an obvious trend that is consistent with expansion, with galaxies that we calculated as being further away also having higher redshifts.
- Our linear fit seems to be pretty good based on the eye test, so our results also seem to be consistent with a linear model for expansion as well.
- Our hubble constant however, is a bit low, with a calculated value of 58.6, which is much lower than excepted values, which tend to be in the range of 65-75. this may be attributed, however, to several, rather large sources of error, which will be discussed in the next part of this lab report.
- One large source of error that we noticed is the fact that most of our galaxies occupy a pretty tight range of redhsifts, but one of our galaxies, NGC 3516, has a much larger redshift. Thus, when conducting a linear fit, we are essentially just fitting the rest of the data points to that single data point. Therefore, our measurement of the angular size of this galaxy has much more weight on our overall measurement than any of our other galaxies, which is not ideal. If the value we have for angular size of this galaxy is changed just a little bit, we get a completely different result for our hubble constant. I tried to correct for this by adding more galaxies, which definitely made this effect have less of an impact, but it still represents a rather large potential source of error. For example, if ngc 3516 is measured to be .0275 instead of ~.025, an error of just 9%, the Hubble constant jumps from 58.6 to 64.6. Our bad measurement for the hubble constant may thus be at least partially atributed to the sensitivity of this specific data point.

Part Three: Uncertainty and Error Analy

- There are certainly several large sources of stastical uncertainty. First of all, the assumption that galaxies are all the same size is not true, and rather would follow some distribution that peaks at 22 kpc but occupies a wide range. Additionally, there is stastical error present in the measurements of redshift as well, although those uncertanties are small, and are thus likely negligible when compared to the statistical uncertanties related to galaxy size. An example of this is m87, which is much larger than 22 kpc (it is in fact around 40.5 kpc), and thus appears larger on the sky despite having a relatively low redshift (and m87 is indeed the most discrepant point, well above our model line, which can be attributed to this fact).
- However, even if we increased the number of galaxies, while the linear fit may get better, it certainly would not shift to the amount of 12 km/s/Mpc, indicating that there must be also be systematics present.
- In terms of systematic error, the largest source is likely in where the cutoff between background and galaxy is present. This source of error is systematic and not stastical, as the same choice was made for all galaxies, and thus all galaxies will be shifted by more or less the same amount by this error. Specifically, based on our calculated hubble constant, there seems to be a systematic shift downward of angular size, where we underestimated all of the angular size of many of the galaxies, as mentioned briefly in the conclusions section of that analysis.
- Given that both of these potential sources of error are rather large, our calculated Hubble constant likely falls within a decently reasonable range (given that accepted values for the hubble constant range from 65-75), even if it is quite low.
- Finally, there is also the potential for both systematic and stastical errors within my measurements themselves, as my vectors couldve been drawn slightly differently in each galaxy (though I tried to keep my technique at the very least consistent across all of the galaxies), but those errors would likely be very small in comparison with the other sources of error I have already mentioned.

Part Four: Extras and Potential Improvei

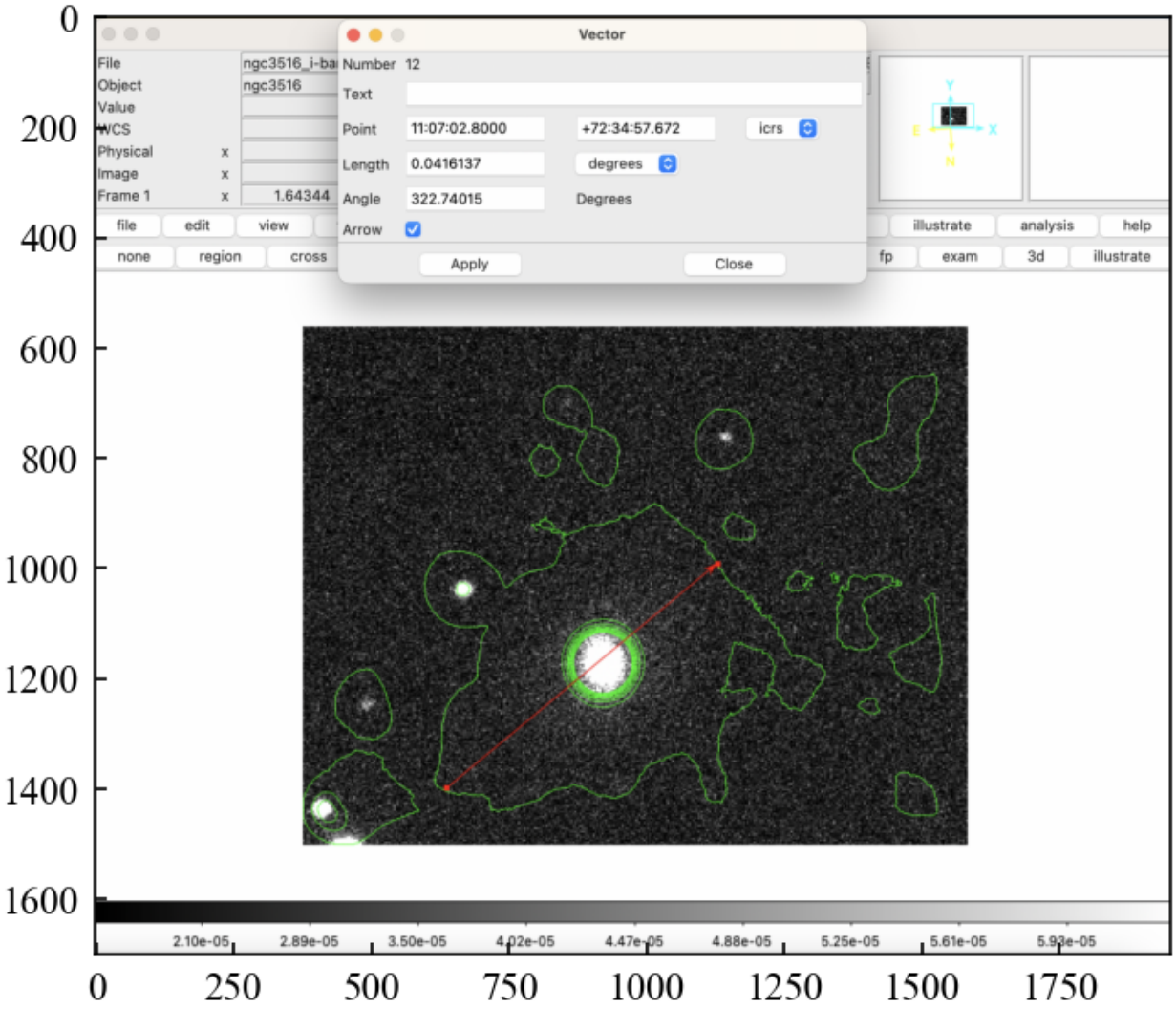
I tried a lot of different methods in this lab, with varying success, before finally settling on the method I ended up with, so I thought I would showcase a little bit of my thought process as why I chose to present the method I ended up deciding on rather than other methods I used (even though some of them even produced better measurements for the hubble constant)

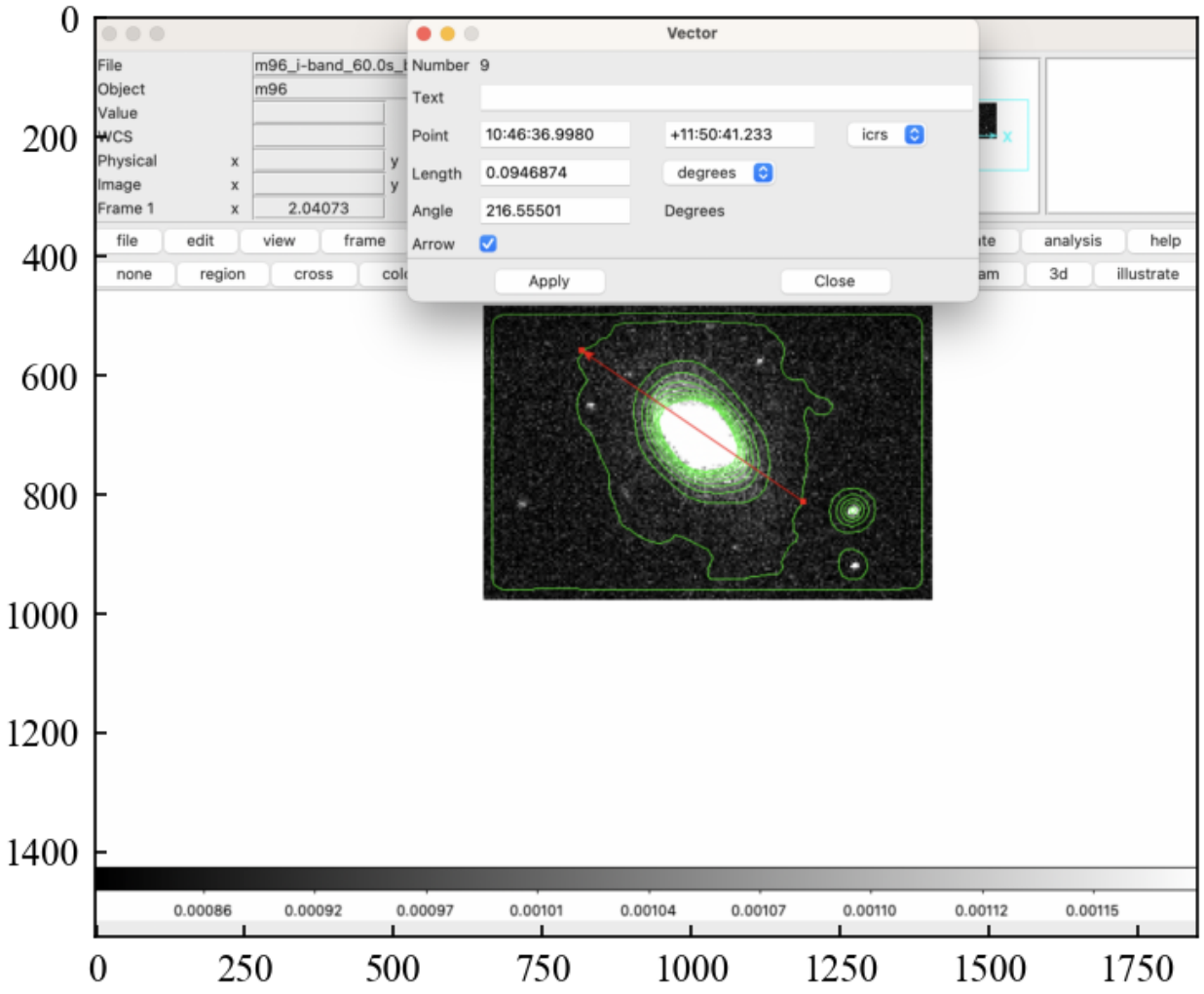
4.1: Experimenting with binning

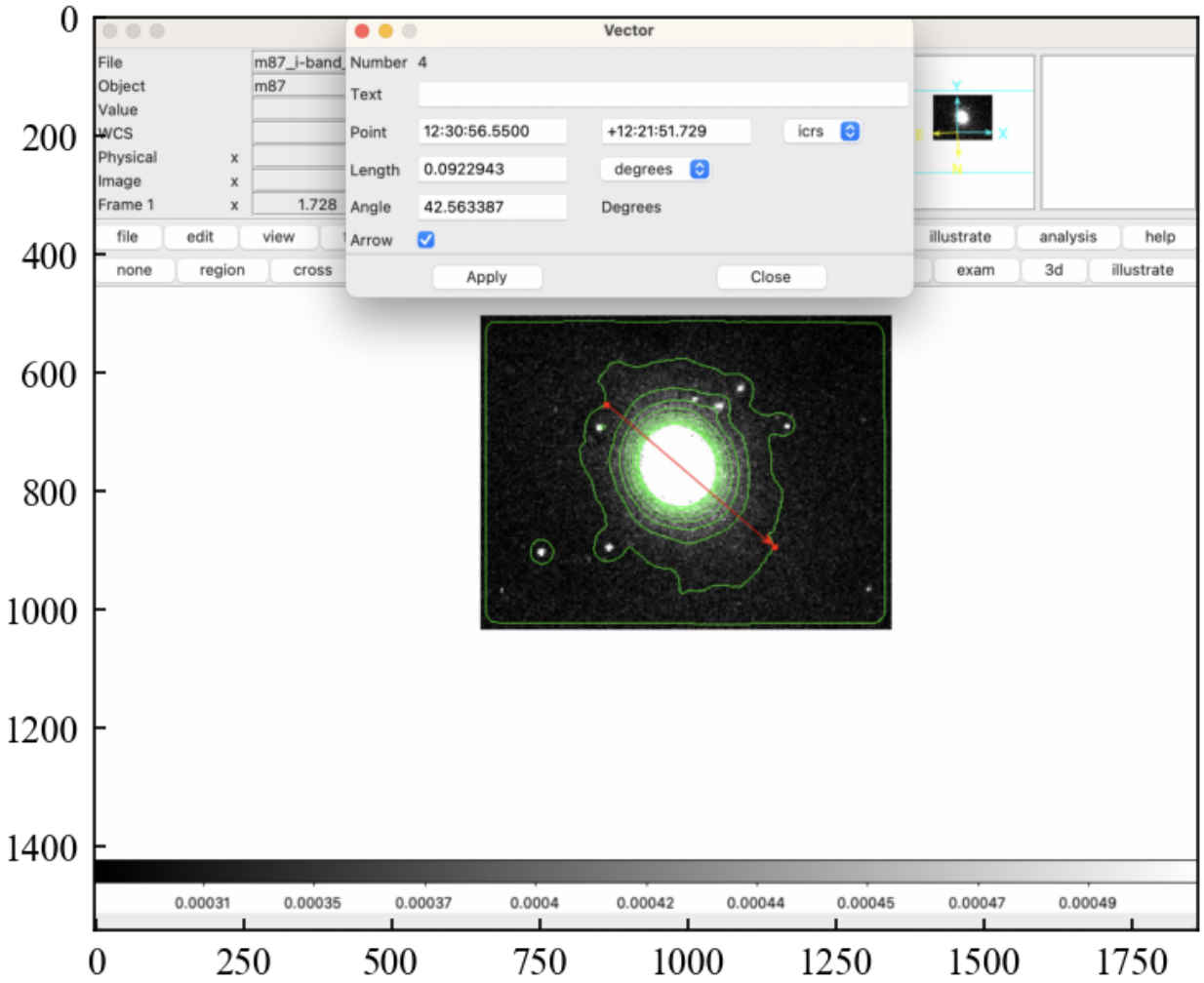
- I tried to use ds9's binning function to increase by signal to noise, with varying success.
- Specifically, I noticed that ds9's binning function was inconsistent in how it increased signal to noise. For some galaxies, I noticed a sharp increase in angular size, for others, I noticed small to no change, and for a few I even saw a decrease. Thus I decided that at least based on my current knowledge of binning, using scaling to increase brightness and create a more sharp cutoff between image and background was a more stable solution. I do think that with more advanced software or knowledge about binning, this method could produce a better result, as it did seem to do a decent job correcting for the systematic under-approximation of angular size discussed in section 3 of this report.
- Examples and conclusions drawn from those examples are examined below.
- Finally, the contour function seemed to do a much worse job at defining where the light started and ended, with the light visually seeming to extend much further than the contours reflected when trying to use binning. This was especially prevalent for galaxies whose angular size did not increase after using binning, which again indicates that this method could potentially be very useful if that obstacle could be surpassed.
- Finally, when changing the bin size, you could get the angular size of the galaxies to be basically be arbitrarily large or small, which felt a lot like over fitting to me, especially since the value we are looking for is already known. To avoid this, I used the block 4 setting to conduct my binning analysis in ds9 for all of my galaxies, whether or not the results it produced were ideal.

Some Galaxies Got Much Larger in angular size after binning

In [53]:
```
NGC3516= Image.open("/Users/jackcolvin//NGC 3516 Block.png")
plt.imshow(NGC3516)
plt.show()
M96 = Image.open("/Users/jackcolvin//M96 Block.png")
plt.imshow(M96)
plt.show()
M87 = Image.open("/Users/jackcolvin//M87 Block.png")
plt.imshow(M87)
plt.show()
```


Vector
Number 12
Text
Point 11:07:02.8000 +72:34:57.672 icrs
Length 0.0416137 degrees
Angle 322.74015 Degrees
Arrow
Apply
Close
File
Object
Value
WCS
Physical
Image
Frame 1
ngc3516
1.64344
file edit view
none region cross
illustrate analysis help
fp exam 3d illustrate
2.10e-05 2.89e-05 3.50e-05 4.02e-05 4.47e-05 4.88e-05 5.25e-05 5.61e-05 5.93e-05
0 200 400 600 800 1000 1200 1400 1600
0 250 500 750 1000 1250 1500 1750


Vector
Number 9
Text
Point 10:46:36.9980 +11:50:41.233 icrs
Length 0.0946874 degrees
Angle 216.55501 Degrees
Arrow
Apply
Close
File
Object
Value
WCS
Physical
Image
Frame 1
m96
2.04073
file edit view frame
none region cross
analysis help
3d illustrate
0.00086 0.00092 0.00097 0.00101 0.00104 0.00107 0.00110 0.00112 0.00115
0 200 400 600 800 1000 1200 1400
0 250 500 750 1000 1250 1500 1750

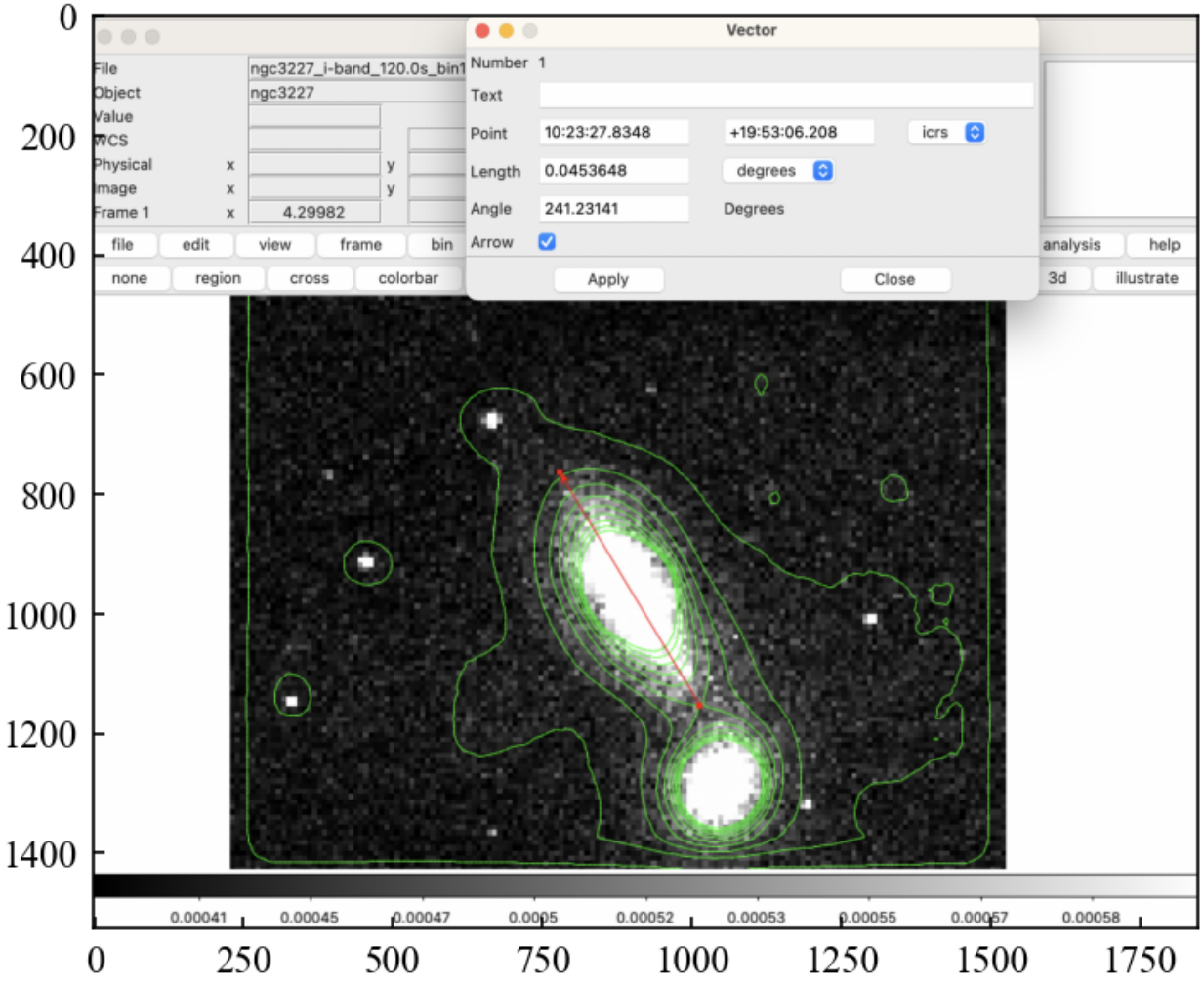


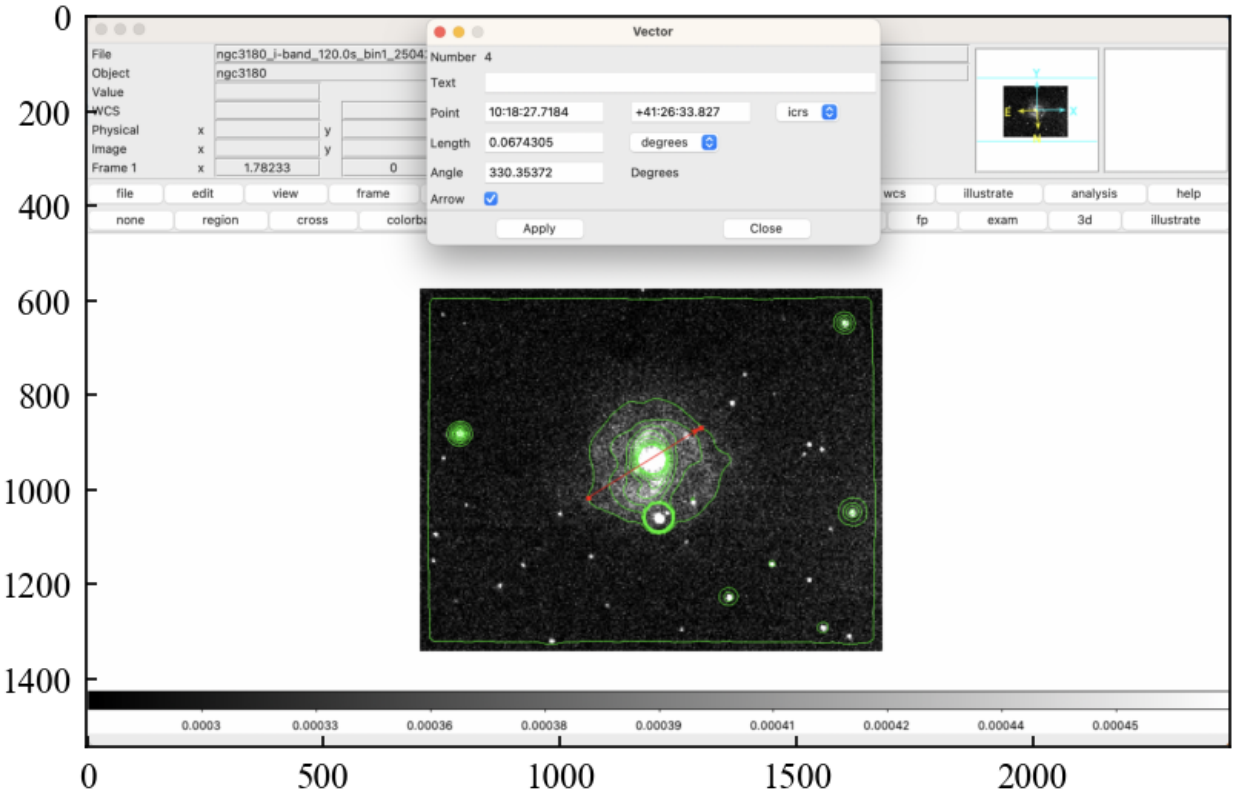
In [54]:
```
M87.close()
M96.close()
NGC3516.close()
```

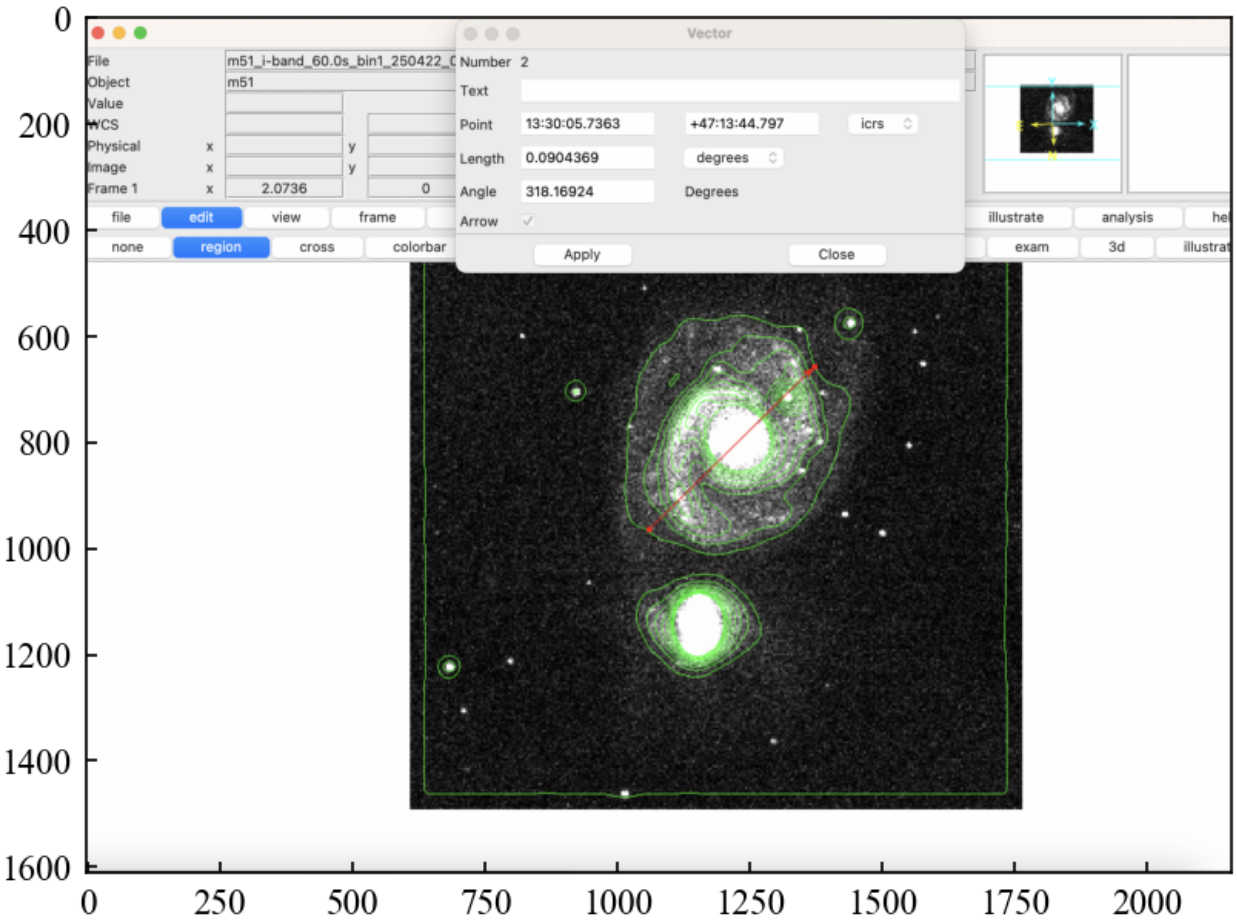
Others stayed very similar in size, and even got a little bit sm

For these galaxies, however, I also noticed that the contours didn't seem to be catching all of the light that was visually eminating from the galaxies. I played around with the parameters a lot, and I could never really get them to correctly reflect the visual extent of the light. I think that without this effect, binning could have been more succesful.

In [13]:
```
NGC3227= Image.open("/Users/jackcolvin//Screenshot 2025-05-05 at 1.00.54 PM.
plt.imshow(NGC3227)
plt.show()
NGC3180= Image.open("/Users/jackcolvin//Screenshot 2025-05-05 at 2.28.10 PM.
plt.imshow(NGC3180)
plt.show()
M51= Image.open("/Users/jackcolvin//Screenshot 2025-05-05 at 5.05.08 PM.png"
plt.imshow(M51)
plt.show()
```




Vector
Number 4
Text
Point 10:18:27.7184 +41:26:33.827 icrs
Length 0.0674305 degrees
Angle 330.35372 Degrees
Arrow
Apply
Close
File
Object
Value
WCS
Physical
Image
Frame 1
ngc3180
file edit view frame
none region cross
wcs illustrate analysis help
fp exam 3d illustrate
0.0003 0.00033 0.00036 0.00038 0.00039 0.00041 0.00042 0.00044 0.00045
0 200 400 600 800 1000 1200 1400
0 500 1000 1500 2000

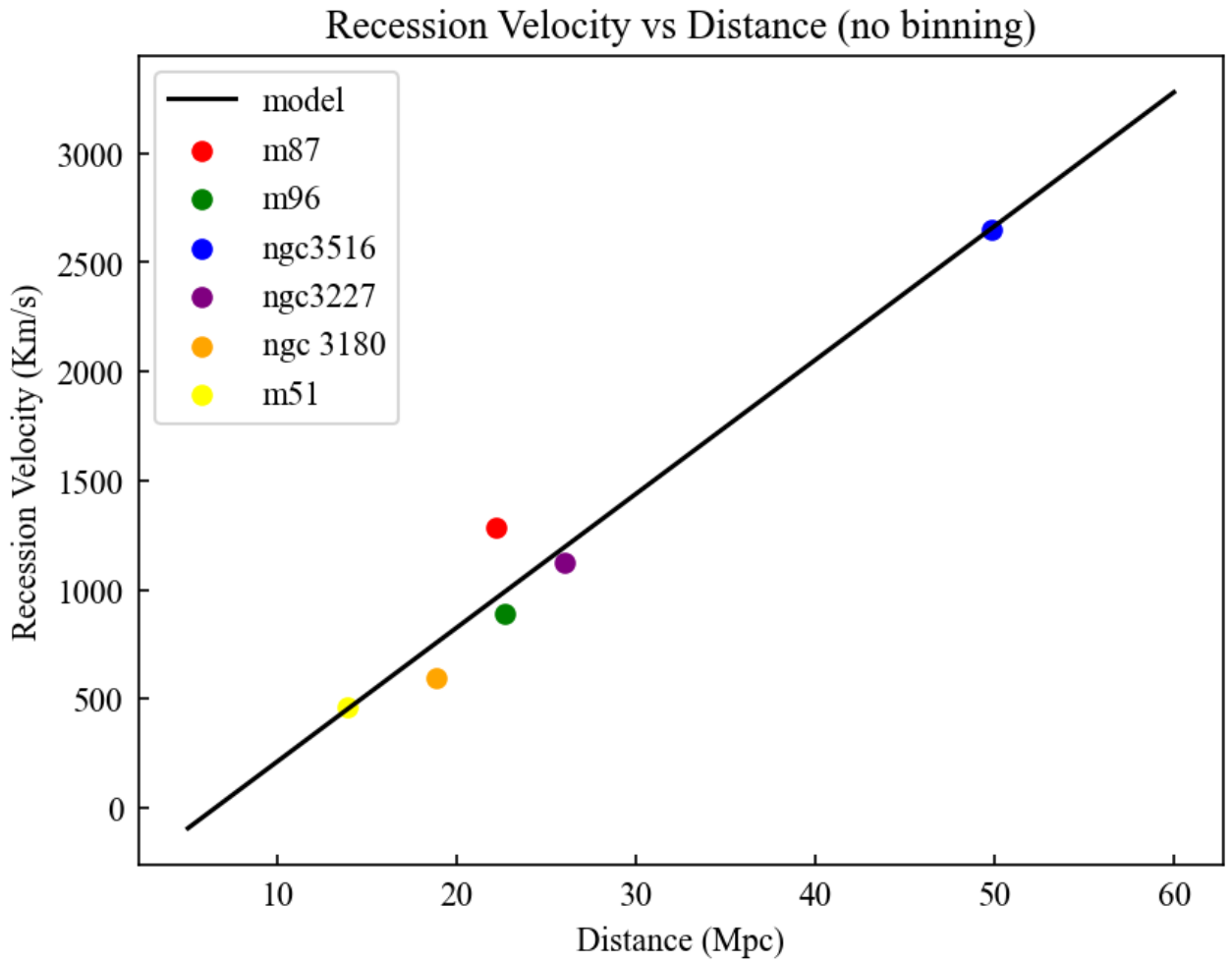

Vector
Number 2
Text
Point 13:30:05.7363 +47:13:44.797 icrs
Length 0.0904369 degrees
Angle 318.16924 Degrees
Arrow
Apply
Close
File
Object
Value
WCS
Physical
Image
Frame 1
m51
file edit view frame
none region cross colorbar
illustrate analysis
exam 3d
0 200 400 600 800 1000 1200 1400 1600
0 250 500 750 1000 1250 1500 1750 2000

In [14]:
```
NGC3180.close()
NGC3227.close()
M51.close()
```

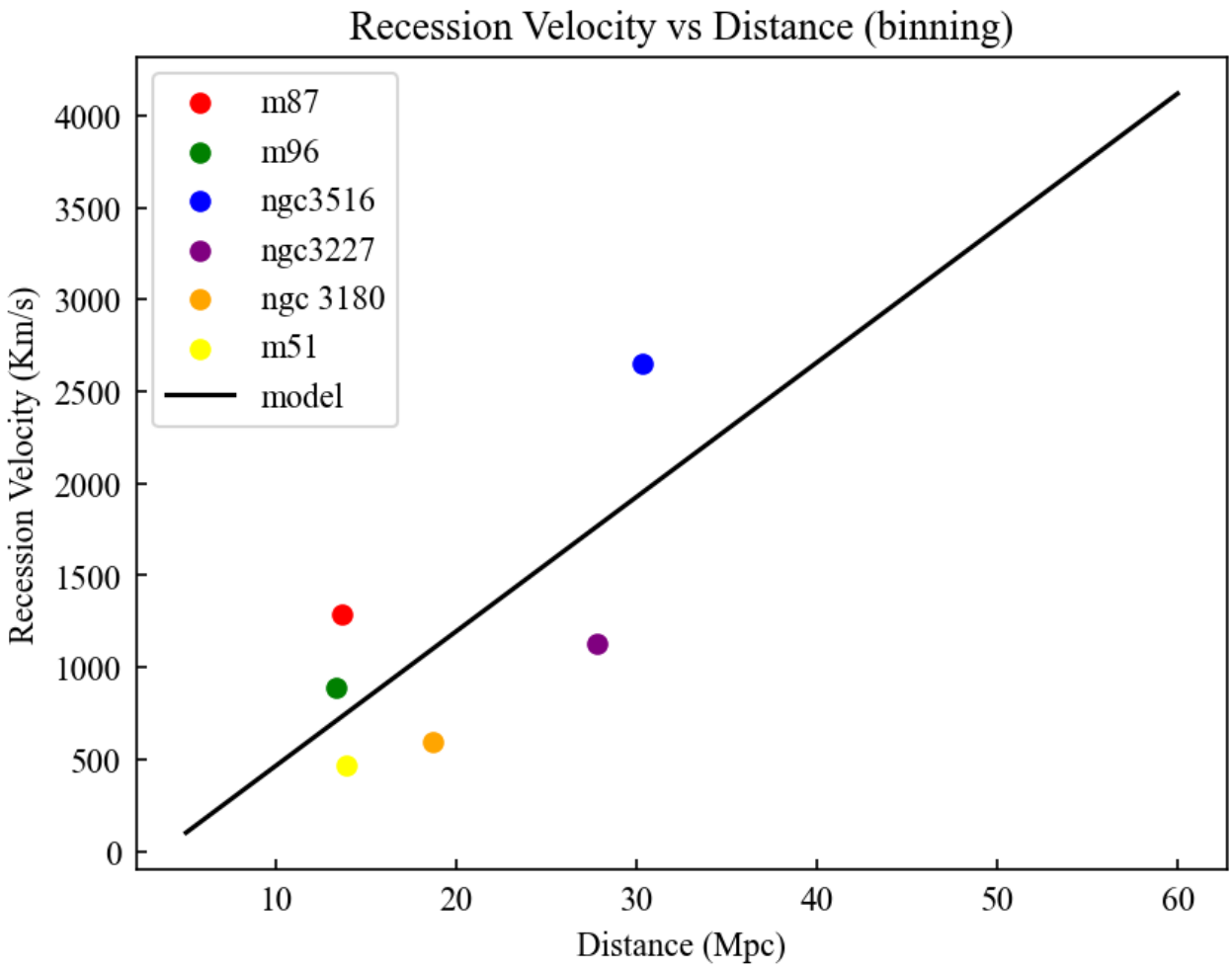
To really show how much this discrepancy skewed my data, I plotted the original positions of these 6 galaxies vs a new fit with binning to show the difference, using the same color dot to represent the same galaxy with the old vs the new measurements

In [111…
```
from scipy import stats
degrees0 = np.array([.0568, .0556, .0253, .0484, .0670, .0907])

radians0 = degrees0 * np.pi/180

distance0 = 22000/radians0

distance0 = distance0 * 1e-6

degrees1 =  np.array([.0922, .0947, .0416, .0454, .0674, .0904])

radians1 = degrees1 * np.pi/180

distance1 = 22000/radians1

distance1 = distance1 * 1e-6

redshift = np.array([.004283, .002967, .008836, .003756, .001975, .001543])

velocity = 300000 * redshift

a0, b0 = np.polyfit(distance0, velocity, 1)
x0 = np.linspace(5, 60, 55)
plt.plot(x0, a0*x0+b0, color = 'black')
plt.scatter(distance0[0], velocity[0], color = 'red')
plt.scatter(distance0[1], velocity[1], color = 'green')
plt.scatter(distance0[2], velocity[2], color = 'blue')
plt.scatter(distance0[3], velocity[3], color = 'purple')
plt.scatter(distance0[4], velocity[4], color = 'orange')
plt.scatter(distance0[5], velocity[5], color = 'yellow')
plt.title("Recession Velocity vs Distance (no binning)")
plt.xlabel("Distance (Mpc)")
plt.ylabel("Recession Velocity (Km/s)")
plt.legend(["model", "m87", "m96", "ngc3516", "ngc3227", "ngc 3180", "m51"])
plt.show()
print(f'Our calculated hubble constant is {a0:.1f}')

plt.scatter(distance1[0], velocity[0], color = 'red')
plt.scatter(distance1[1], velocity[1], color = 'green')
plt.scatter(distance1[2], velocity[2], color = 'blue')
plt.scatter(distance1[3], velocity[3], color = 'purple')
plt.scatter(distance1[4], velocity[4], color = 'orange')
plt.scatter(distance0[5], velocity[5], color = 'yellow')
```

```
plt.title("Recession Velocity vs Distance (binning)")
plt.xlabel("Distance (Mpc)")
plt.ylabel("Recession Velocity (Km/s)")
a1, b1 = np.polyfit(distance1, velocity, 1)
x1 = np.linspace(5, 60, 55)
plt.plot(x1, a1*x1+b1, color = 'black')
plt.legend(["m87", "m96", "ngc3516", "ngc3227", "ngc 3180", "m51", "model"])
plt.show()
print(f'Our calculated hubble constant is {a1:.1f}')
```




```
Our calculated hubble constant is 61.3
```



Our calculated hubble constant is 73.1

4.1 Conclusions on Binning results :

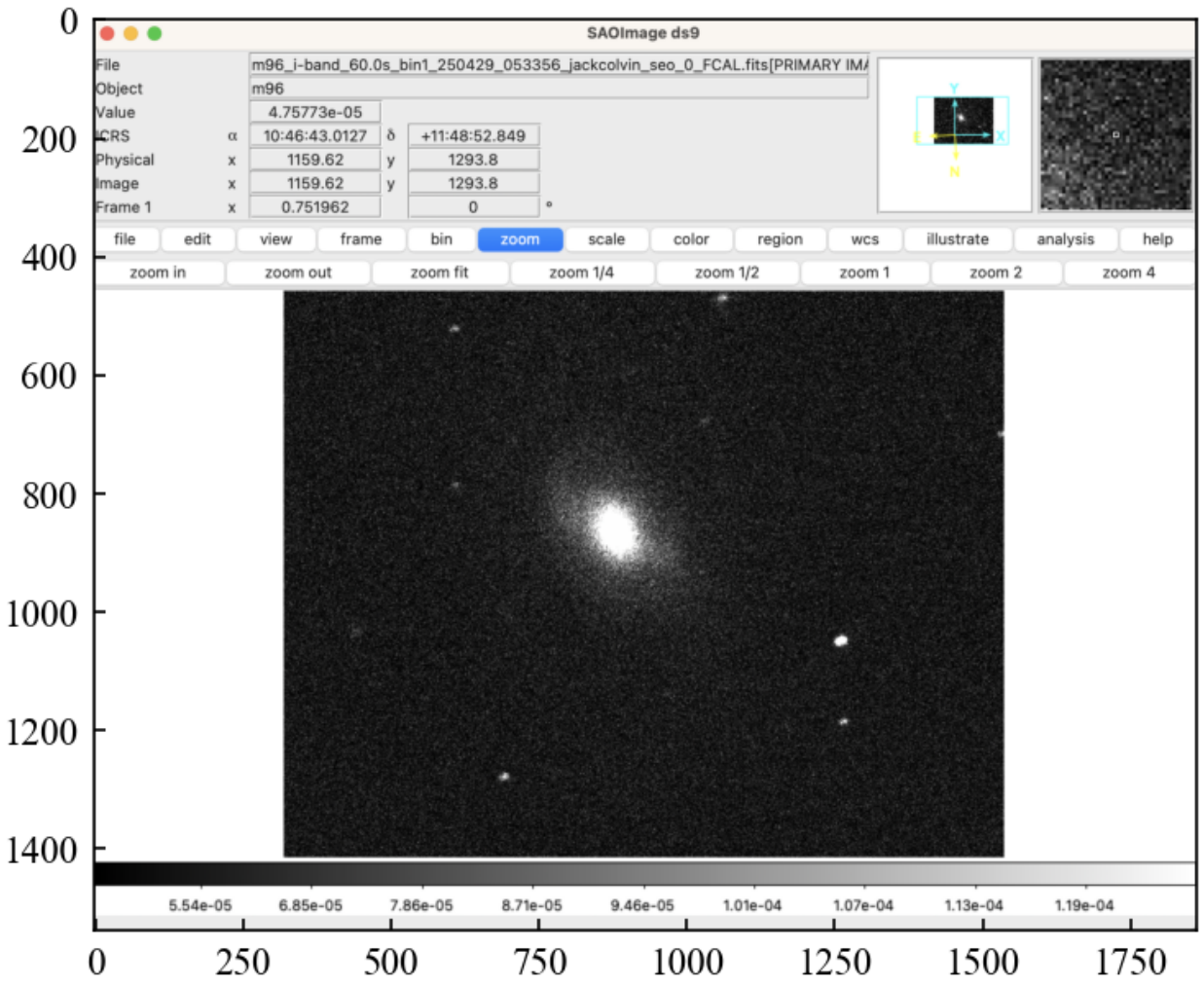
- As seen in the graphs, the dots occupy completely different positions before and after binning, which to me indicated that binning did not have a consistent effect on angular size and rather had a much more random effect. This meant that binning itself was not a good method to correct for the systematic under-estimation of angular size that I was getting from my original method
- The linear fit was also much worse in the binning case. Thus, although the value I got for the hubble constant itself was better, I decided to stick with my original method as it gave much more consistent results, even if it was susceptible to statistical and systematic errors that were making it innacurate.
- I do think, however, that binning did a good job increasing signal to noise and I did get a "better" value for my hubble constant, indicating that there is certainly potential for this method to be potentially a better method to use than my original method, if I found a way to get consistent results across different galaxies.
- Interestingly, our hubble constant using our original method is actually better (61.3) than our orginal calculated value (58.5), further proving that there is some systematic effect present, as if the error was purely statistical, the number should get better, not worse, when we add more galaxies.

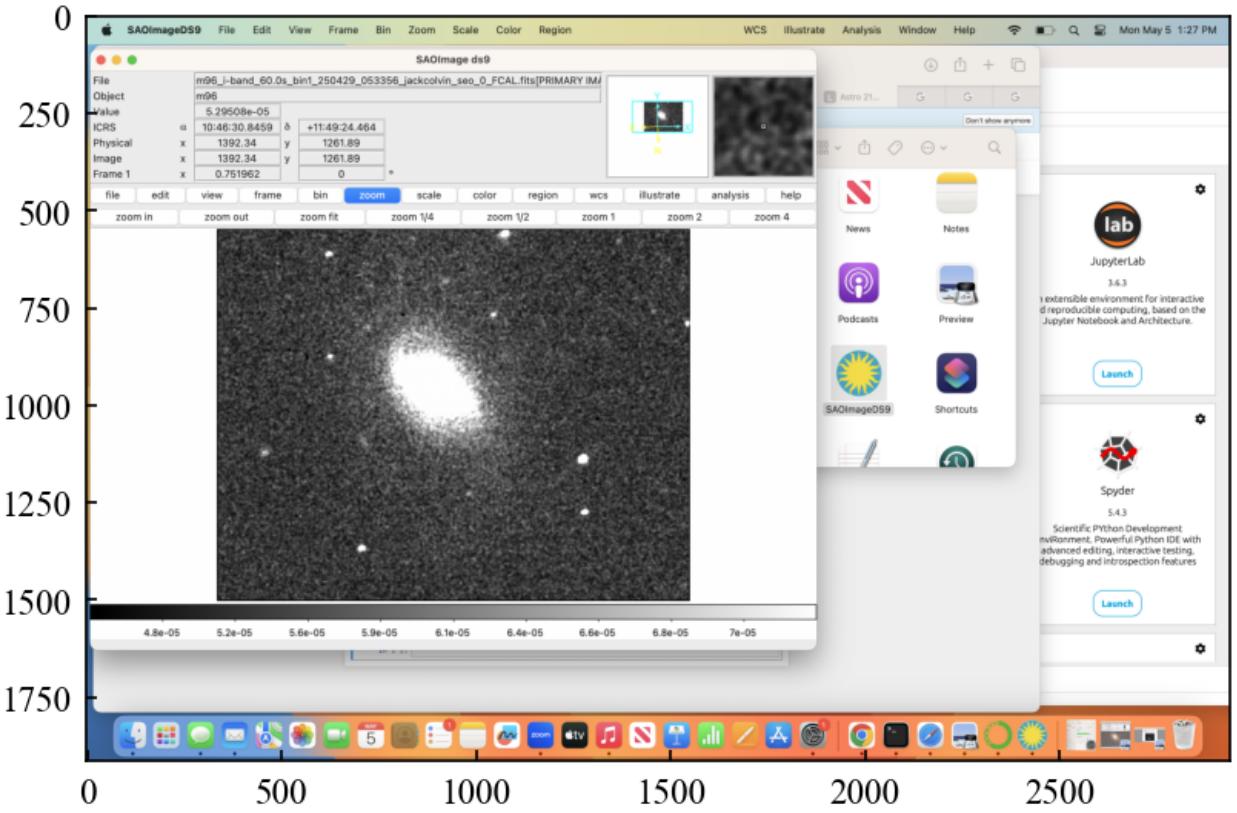
4.2: Experimenting with smoothing

Another way I very briefly looked at increasing signal to noise was using the smoothing feature in ds9, however I discared this method rather quickly, as it just seemed to arbitrarily make the light from the galaxy expand further. It also simply decreased the resolution of the galaxies, making it much harder to determine the line between galaxy and background. The background itself also brightened significantly, which further messed up the contours, making them very jagged even on the same smoothness I used for binning and my normal analysis. This can be seen in the range of pixel values, which was an order of magnitude smaller than for the unsmoothed images. Thus, while it did increase signal to noise, it did so by much less than the binning becaused it increased the noise along with the signal. Thus, I decided to discard this method as well without as much analysis as with binning (I just looked at my 4 observations qualitatively to determine that this method was likely not a good strategy). If I were to extend this lab, I would certainly look much more into binning rather than smoothing as a way to increase signal to noise.

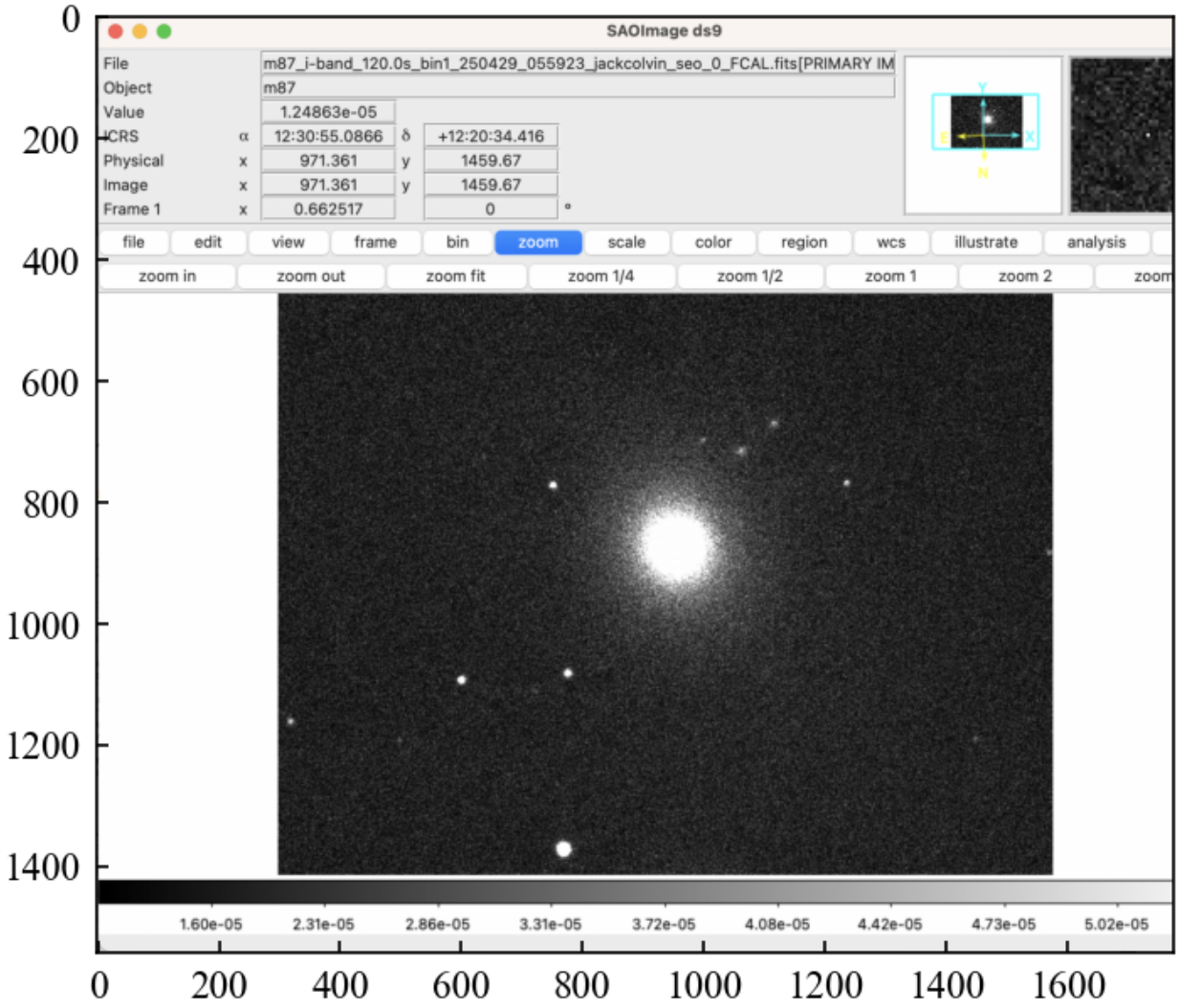
Smoothing Images!!

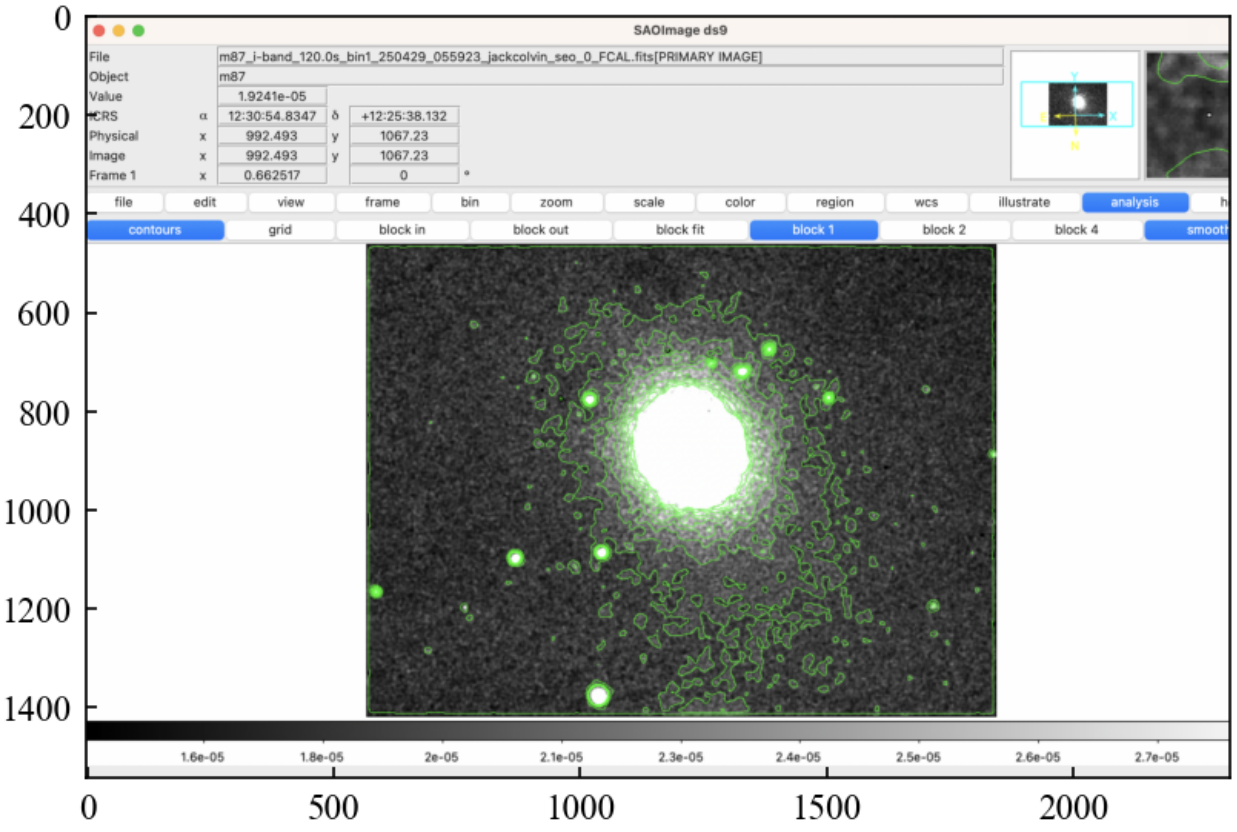
In [79]:
```
M96_= Image.open("/Users/jackcolvin//Screenshot 2025-05-05 at 1.27.29 PM.png
plt.imshow(M96_)
plt.show()
M96_smooth= Image.open("/Users/jackcolvin//Screenshot 2025-05-05 at 1.27.51
plt.imshow(M96_smooth)
plt.show()
```



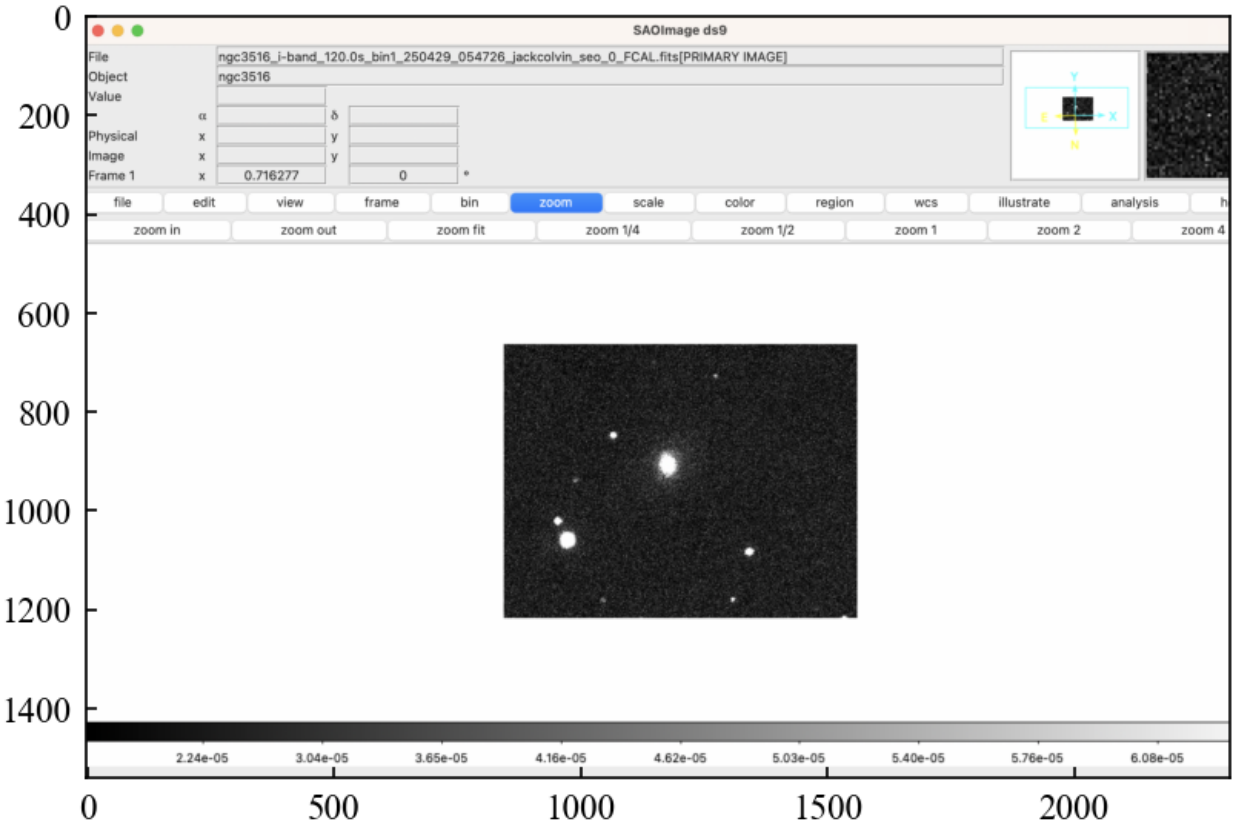


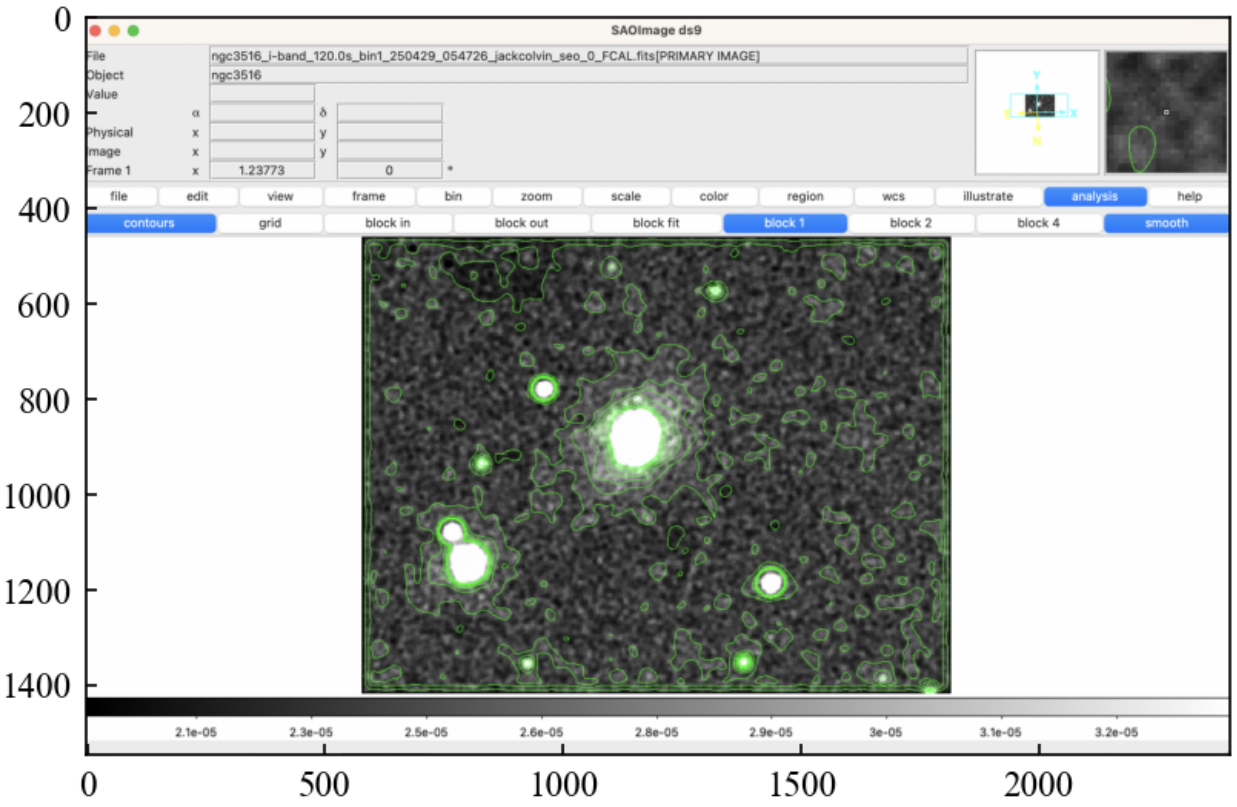
In [80]:
```
M87_= Image.open("/Users/jackcolvin//M87 blank.png")
plt.imshow(M87_)
plt.show()
M87_smooth= Image.open("/Users/jackcolvin//Screenshot 2025-05-05 at 1.33.48
plt.imshow(M87_smooth)
plt.show()
```


SAOImage ds9
File
m87_j-band_120.0s_bin1_250429_055923_jackcolvin_seo_0_FCAL.fits[PRIMARY IM
Object
m87
Value
1.24863e-05
WCS
α
12:30:55.0866
δ
+12:20:34.416
Physical
x
971.361
y
1459.67
Image
x
971.361
y
1459.67
Frame 1
x
0.662517
0
°
file
edit
view
frame
bin
zoom
scale
color
region
wcs
illustrate
analysis
zoom in
zoom out
zoom fit
zoom 1/4
zoom 1/2
zoom 1
zoom 2
1.60e-05
2.31e-05
2.86e-05
3.31e-05
3.72e-05
4.08e-05
4.42e-05
4.73e-05
5.02e-05
0
200
400
600
800
1000
1200
1400
1600

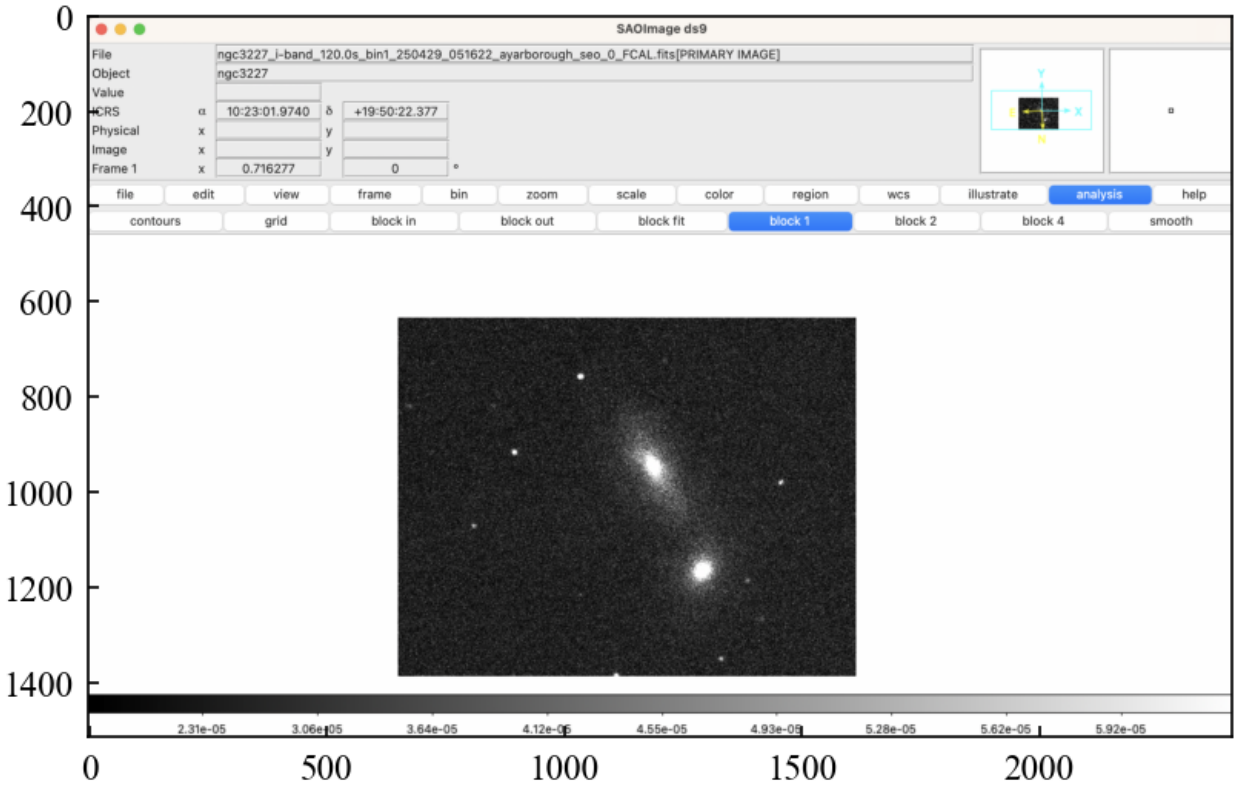


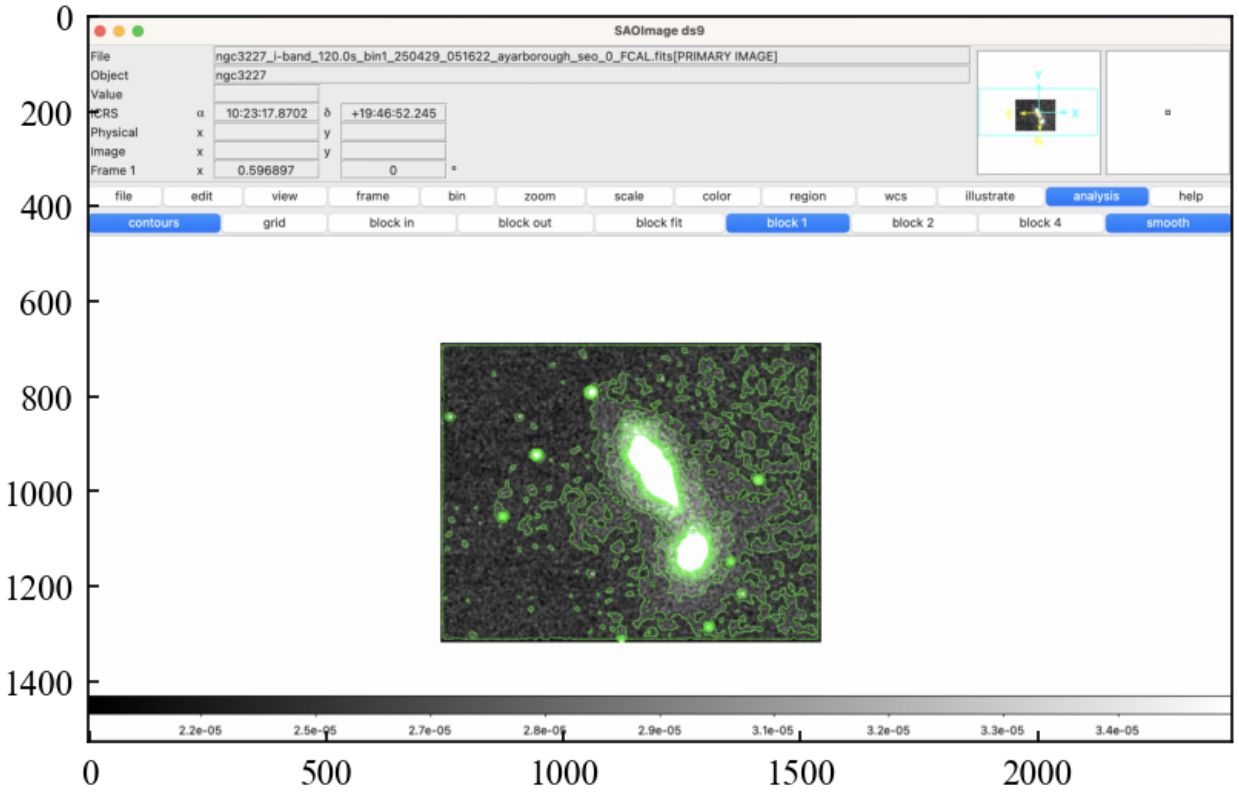
In [81]:
```
NGC3516_= Image.open("/Users/jackcolvin/Screenshot 2025-05-05 at 1.34.53 PM.
plt.imshow(NGC3516_)
plt.show()
NGC3516_smooth= Image.open("/Users/jackcolvin//Screenshot 2025-05-05 at 1.35
plt.imshow(NGC3516_smooth)
plt.show()
```


SAOImage ds9
File
ngc3516_i-band_120.0s_bin1_250429_054726_jackcolvin_seo_0_FCAL.fits[PRIMARY IMAGE]
Object
ngc3516
Value
Physical
Image
Frame 1
0.716277
file edit view frame bin zoom scale color region wcs illustrate analysis
zoom in zoom out zoom fit zoom 1/4 zoom 1/2 zoom 1 zoom 2 zoom 4
2.24e-05 3.04e-05 3.65e-05 4.16e-05 4.62e-05 5.03e-05 5.40e-05 5.76e-05 6.08e-05
0 200 400 600 800 1000 1200 1400
0 500 1000 1500 2000


SAOImage ds9
File
ngc3516_i-band_120.0s_bin1_250429_054726_jackcolvin_seo_0_FCAL.fits[PRIMARY IMAGE]
Object
ngc3516
Value
Physical
Image
Frame 1
1.23773
file edit view frame bin zoom scale color region wcs illustrate analysis help
contours grid block in block out block fit block 1 block 2 block 4 smooth
2.1e-05 2.3e-05 2.5e-05 2.6e-05 2.8e-05 2.9e-05 3e-05 3.1e-05 3.2e-05
0 200 400 600 800 1000 1200 1400
0 500 1000 1500 2000

In [82]:
```
NGC3227_= Image.open("/Users/jackcolvin//Screenshot 2025-05-05 at 1.39.05 PM
plt.imshow(NGC3227_)
plt.show()
NGC3227_smooth= Image.open("/Users/jackcolvin//Screenshot 2025-05-05 at 1.40
plt.imshow(NGC3227_smooth)
plt.show()
```





In [83]:
```
NGC3227_.close()
NGC3227_smooth.close()
NGC3516_.close()
NGC3516_smooth.close()
M87_.close()
M87_smooth.close()
M96_.close()
M96_smooth.close()
```

4.2 Conclusions on Smoothing :

- Generally, the contours were very messy, and it would be very difficult to get a good measure of angular size
- However, just like binning, there could be potential in this method. It does increase signal to noise, and I did only try this method with a couple sets of parameters. Thus, maybe by trying more different parameter settings, it could be able to increase signal to noise without also increasing the brightness of the background and without making contours very difficult to draw, in which case it would be a very useful method.

Part 5: Overall Lab Conclusions!

- I had a lot of fun trying different methods to see what worked and what didn't in this lab. Although my hubble constant value wasn't ideal, I got a very reasonable linear fit which provided clear evidence of an expanding universe.
- My results also indicated a systematic source of error, specifically a systematic underestimation of angular size. I tried using smoothness and binning to offset this error, with mized results. It was very interesting to get an introduction to both of those methods, as I have never used them before, and although they did not work, they definitely have the potential to produce more accurate results if utilized correctly.
- It was also very interesting using ds9 rather than python for this lab. I have learned about the benefits of both programs: specifically that ds9 is maybe easier and quicker to use, but with the cost of it not being able to do even some simple operations, such as subtracting the median noise level from the image, that I had to work around throughout this lab. I think using both tools in tandem is probably the best way to acheive good results.

In [ ]: