final_project_part_2

March 6, 2024

[7]:
```
from google.colab import files
uploaded = files.upload()
```

```
<IPython.core.display.HTML object>
```

```
Saving hlsp_hudf12_hst_wfc3ir_udfmain_f105w_v1.0_drz.fits to
hlsp_hudf12_hst_wfc3ir_udfmain_f105w_v1.0_drz.fits
```

setup

[8]:
```
import numpy as np
```

[9]:
```
pip install sep
```

```
Requirement already satisfied: sep in /usr/local/lib/python3.10/dist-packages
(1.2.1)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages
(from sep) (1.25.2)
```

[10]:
```
import sep
```

extra setup

[11]:
```
from astropy.io import fits
import matplotlib.pyplot as plt
from matplotlib import rcParams

%matplotlib inline

rcParams['figure.figsize'] = [10., 8.]
```

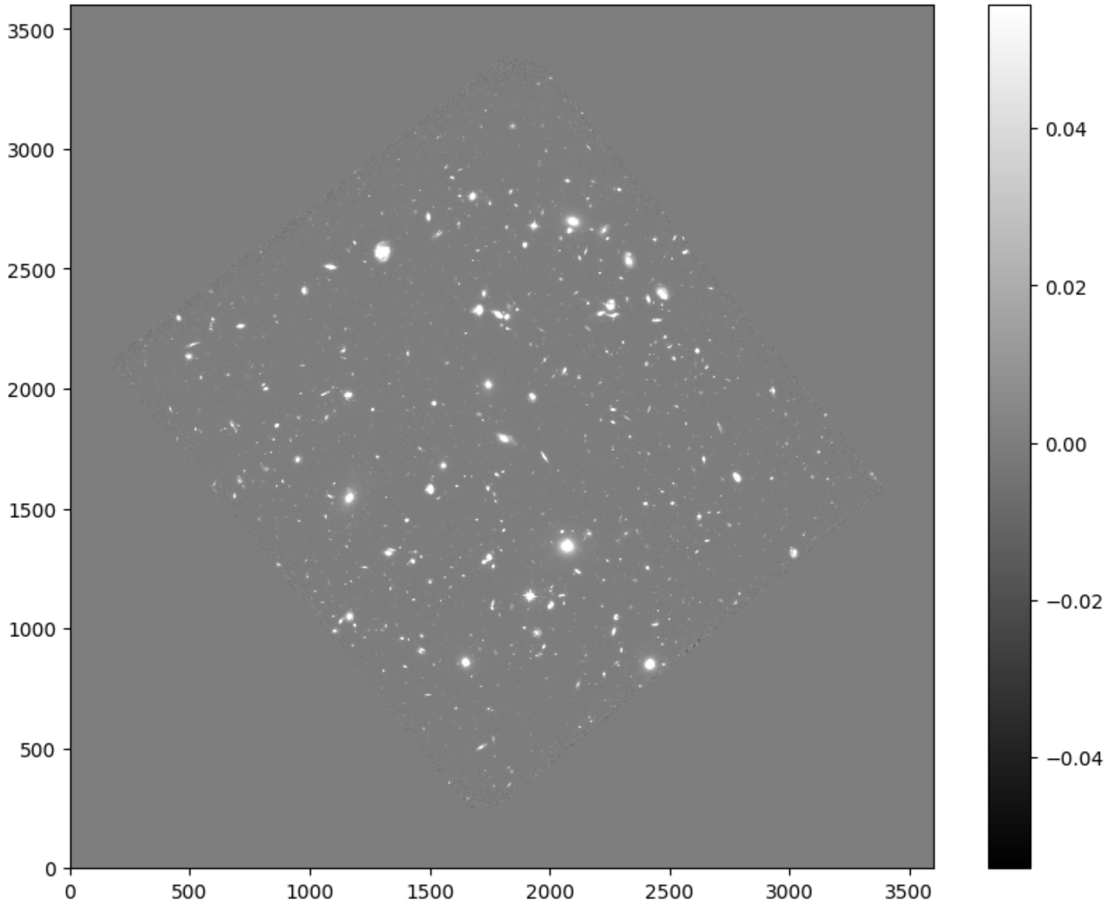
read image

[12]:
```
hdul = fits.open("hlsp_hudf12_hst_wfc3ir_udfmain_f105w_v1.0_drz.fits")
data = hdul[0].data
```

show image

[13]:
```
m, s = np.mean(data), np.std(data)
plt.imshow(data, interpolation='nearest', cmap='gray', vmin=m-s, vmax=m+s,␣
 ↪origin='lower')
```

```
plt.colorbar()
plt.savefig('2dimage.png')
plt.show()
```



measure background

[14]:
```
mask = None
```

[15]:
```
data = data.byteswap().newbyteorder()
```

[16]:
```
bkg = sep.Background(data)
```

[17]:
```
bkg = sep.Background(data, mask=mask, bw=64, bh=64, fw=3, fh=3)
```

get global mean and noise of image background

[18]:
```
print(bkg.globalback)
print(bkg.globalrms)
```

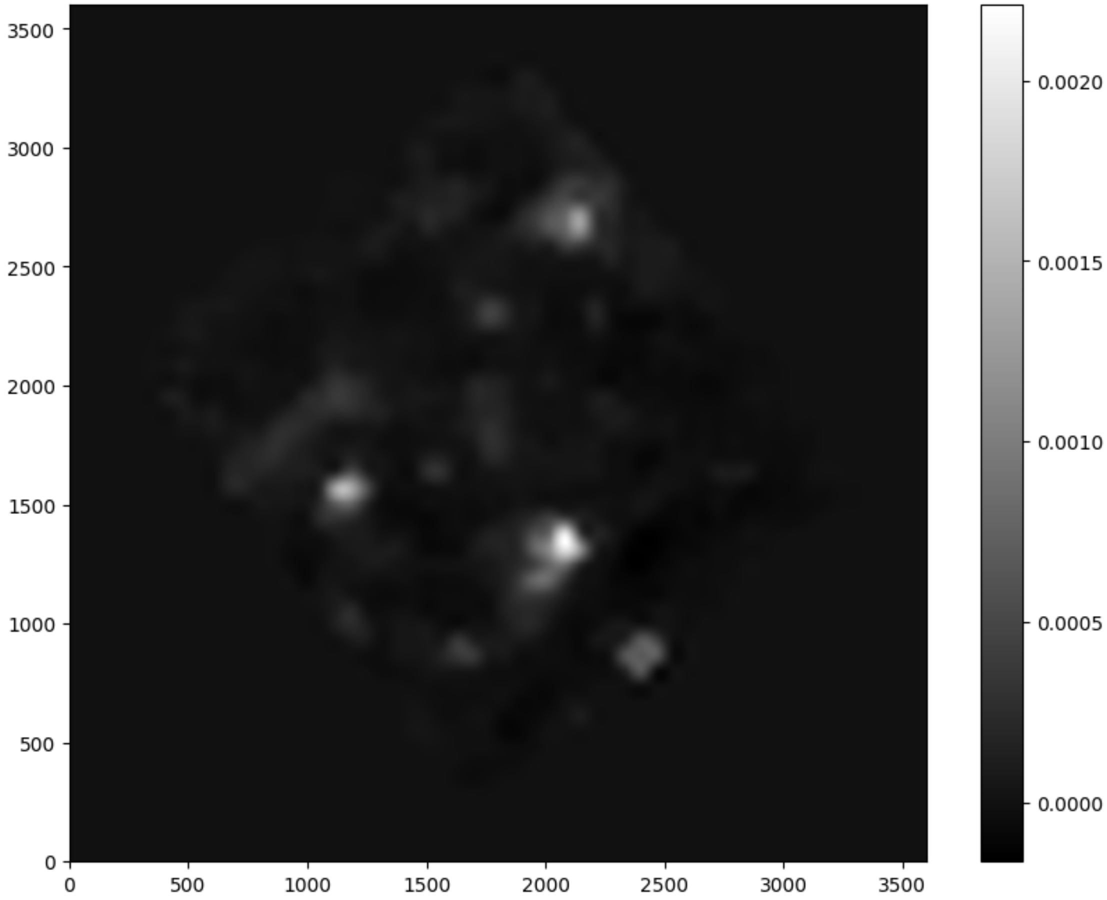
```
0.0
0.0005398219218477607
```

evaluate background

[19]:
```
bkg_image = bkg.back()
```

show background

[20]:
```
plt.imshow(bkg_image, interpolation='nearest', cmap='gray', origin='lower')
plt.colorbar();
plt.savefig('bkg_image.png')
```

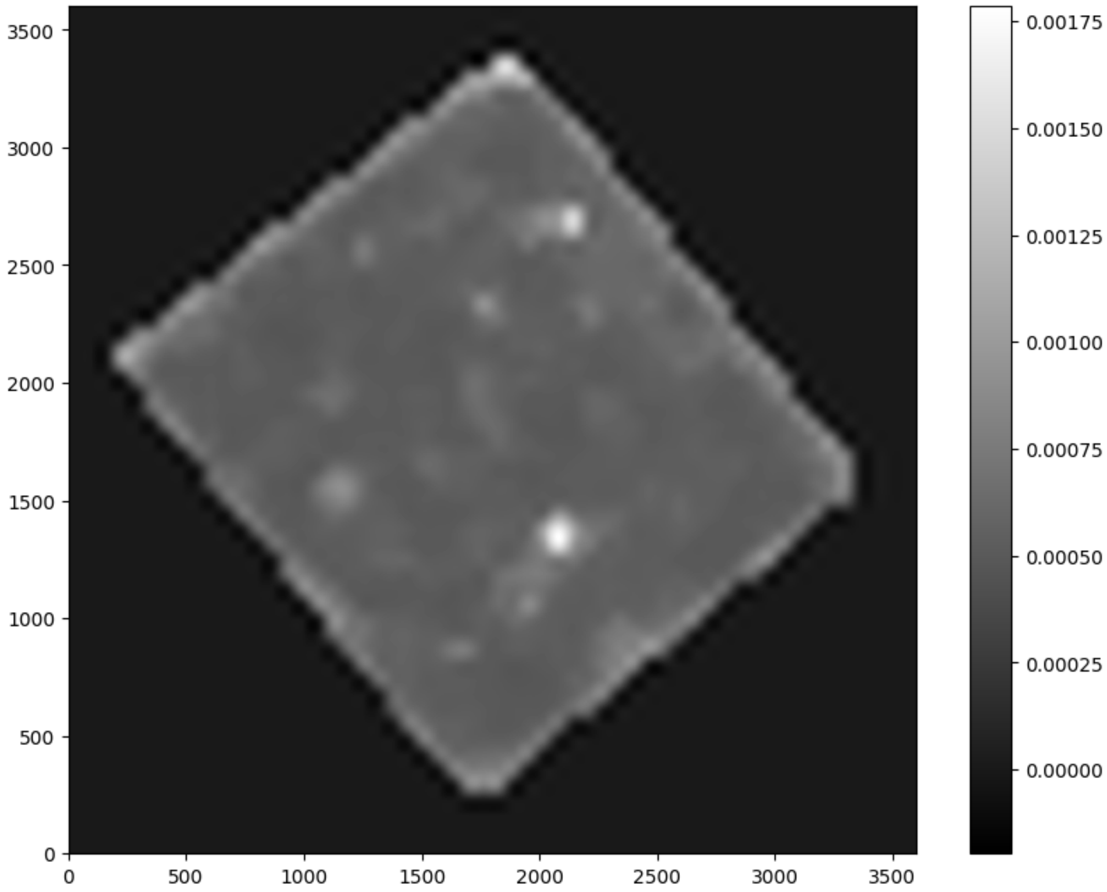


evaluate background

[21]:
```
bkg_rms = bkg.rms()
```

show background noise

[22]:
```
plt.imshow(bkg_rms, interpolation='nearest', cmap='gray', origin='lower')
plt.colorbar();
plt.savefig('bkgnoise_image.png')
```



subtract background

[23]:
```
data_sub = data - bkg
```

object detection

[24]:
```
objects = sep.extract(data_sub, 1.5, err=bkg.globalrms)
```

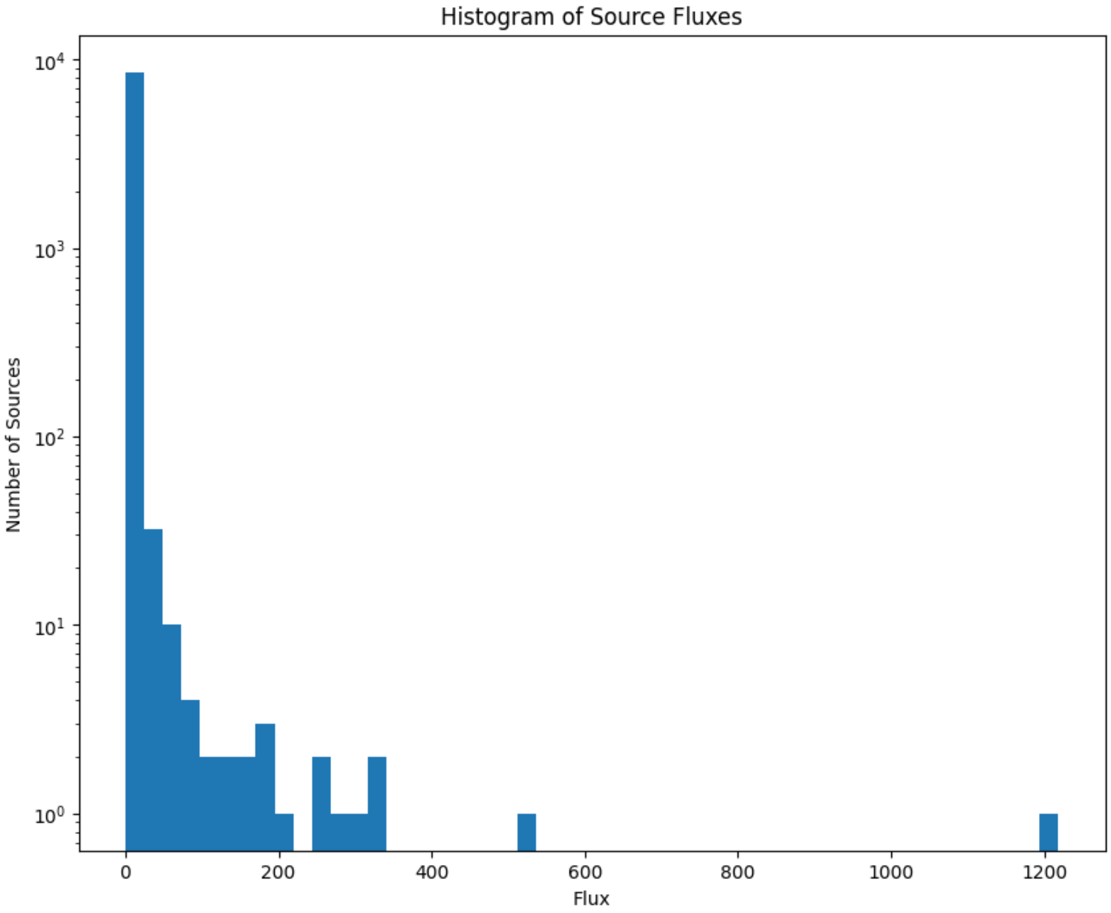
step 6 of assignment

[25]:
```
print("Number of sources:", len(objects))
```

```
Number of sources: 8643
```

histogram of fluxes

[26]:
```
fluxes = objects['flux']
plt.hist(fluxes, bins=50, log=True)
plt.xlabel('Flux')
plt.ylabel('Number of Sources')
plt.title('Histogram of Source Fluxes')
plt.show()
```



step 7 of assignment

mean,median, and standard deviation of the distribution of fluxes

```
[27]: mean_flux = np.mean(fluxes)
median_flux = np.median(fluxes)
std_flux = np.std(fluxes)

print("Mean flux:", mean_flux)
print("Median flux:", median_flux)
print("Standard deviation of fluxes:", std_flux)
```

```
Mean flux: 1.17227448026878
Median flux: 0.0347491130232811
Standard deviation of fluxes: 17.542063555658334
```

find largest outlier in distribution, where it is on the image and how many standard deviations it is away from the mean

[28]:

```
z_scores = (fluxes - mean_flux) / std_flux
largest_outlier_index = np.argmax(np.abs(z_scores))
largest_outlier_flux = fluxes[largest_outlier_index]
num_std_away = np.abs(z_scores[largest_outlier_index])

print("Largest outlier flux:", largest_outlier_flux)
print("Number of standard deviations away from the mean:", num_std_away)
```

```
Largest outlier flux: 1218.4114990234375
Number of standard deviations away from the mean: 69.38973973506889
```

plot background subtracted image

[29]:

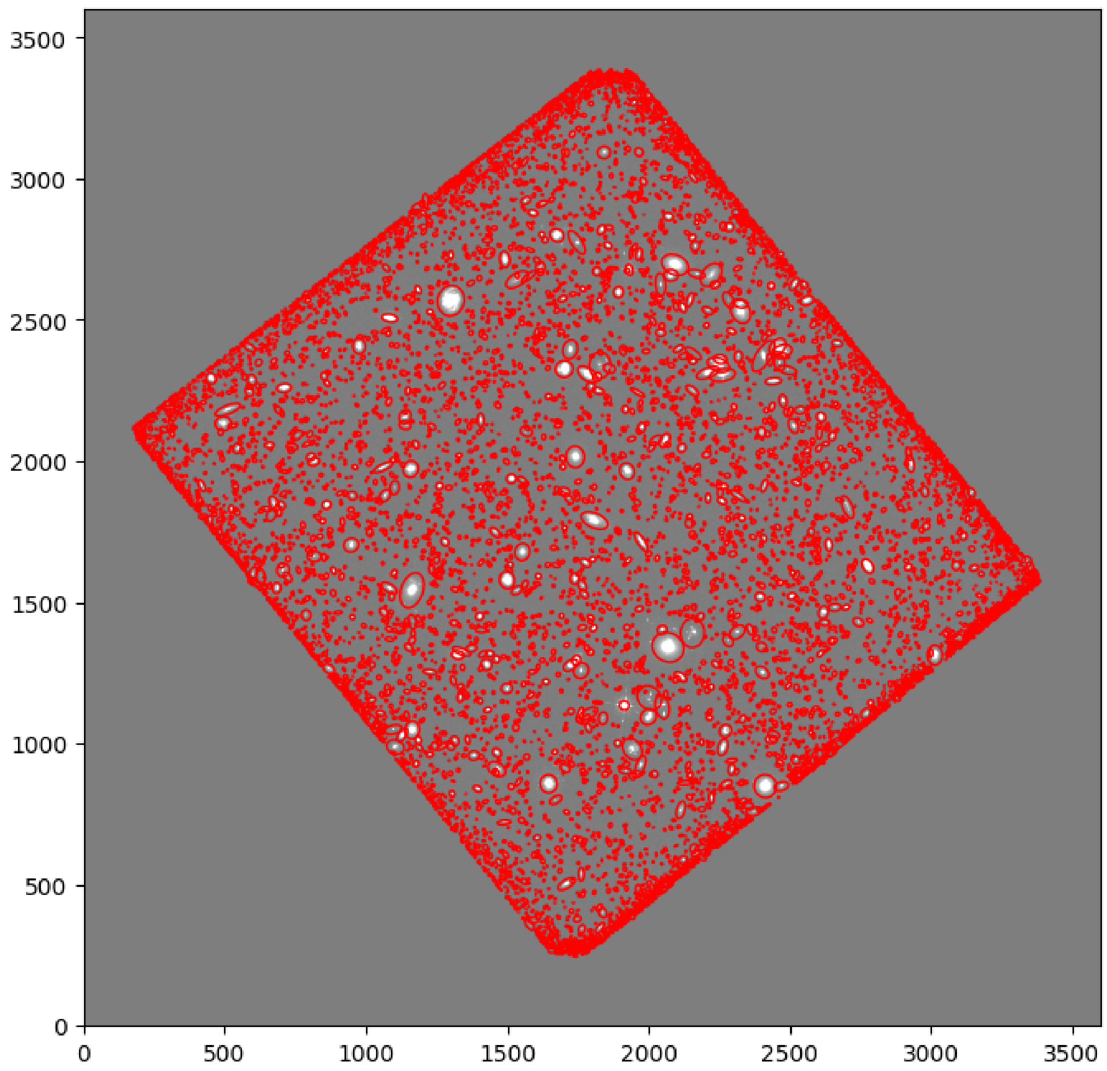
```
from matplotlib.patches import Ellipse
```

[30]:

```
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')

# plot an ellipse for each object
for i in range(len(objects)):
    e = Ellipse(xy=(objects['x'][i], objects['y'][i]),
                width=6*objects['a'][i],
                height=6*objects['b'][i],
                angle=objects['theta'][i] * 180. / np.pi)
    e.set_facecolor('none')
    e.set_edgecolor('red')
    ax.add_artist(e)

plt.savefig('bkgsubtracted_image.png')
```



available fields

[31]:
```
objects.dtype.names
```

[31]:
```
('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')
```

aperture photometry

[32]:

```
flux, fluxerr, flag = sep.sum_circle(data_sub, objects['x'], objects['y'],
                                     3.0, err=bkg.globalrms, gain=1.0)
```

[33]:

```
for i in range(10):
    print("object {:d}: flux = {:f} +/- {:f}".format(i, flux[i], fluxerr[i]))
```

```
object 0: flux = 0.031282 +/- 0.176890
object 1: flux = 0.031018 +/- 0.176142
object 2: flux = -0.024388 +/- 0.002883
object 3: flux = 0.001947 +/- 0.044219
object 4: flux = 0.012457 +/- 0.111649
object 5: flux = -0.011228 +/- 0.002875
object 6: flux = 0.029368 +/- 0.171394
object 7: flux = -0.009126 +/- 0.002875
object 8: flux = 0.048023 +/- 0.219161
object 9: flux = 0.027840 +/- 0.166877
```

step 8 of assignment

[34]:

```
from google.colab import files
uploaded = files.upload()
```

```
<IPython.core.display.HTML object>
```

```
Saving hlsp_hudf12_hst_wfc3ir_udfmain_f160w_v1.0_drz.fits to
hlsp_hudf12_hst_wfc3ir_udfmain_f160w_v1.0_drz.fits
```

```
Saving hlsp_hudf12_hst_wfc3ir_udfmain_f125w_v1.0_drz.fits to
hlsp_hudf12_hst_wfc3ir_udfmain_f125w_v1.0_drz.fits
```

load fits image

[63]:

```
f160w_data = fits.getdata('hlsp_hudf12_hst_wfc3ir_udfmain_f160w_v1.0_drz.fits')
f125w_data = fits.getdata('hlsp_hudf12_hst_wfc3ir_udfmain_f125w_v1.0_drz.fits')
f105w_data = fits.getdata('hlsp_hudf12_hst_wfc3ir_udfmain_f105w_v1.0_drz.fits')
```

normalize pixel values

[64]:

```
f160w_data = f160w_data / np.max(f160w_data)
f125w_data = f125w_data / np.max(f125w_data)
f105w_data = f105w_data / np.max(f105w_data)
```

create false color image

[65]:

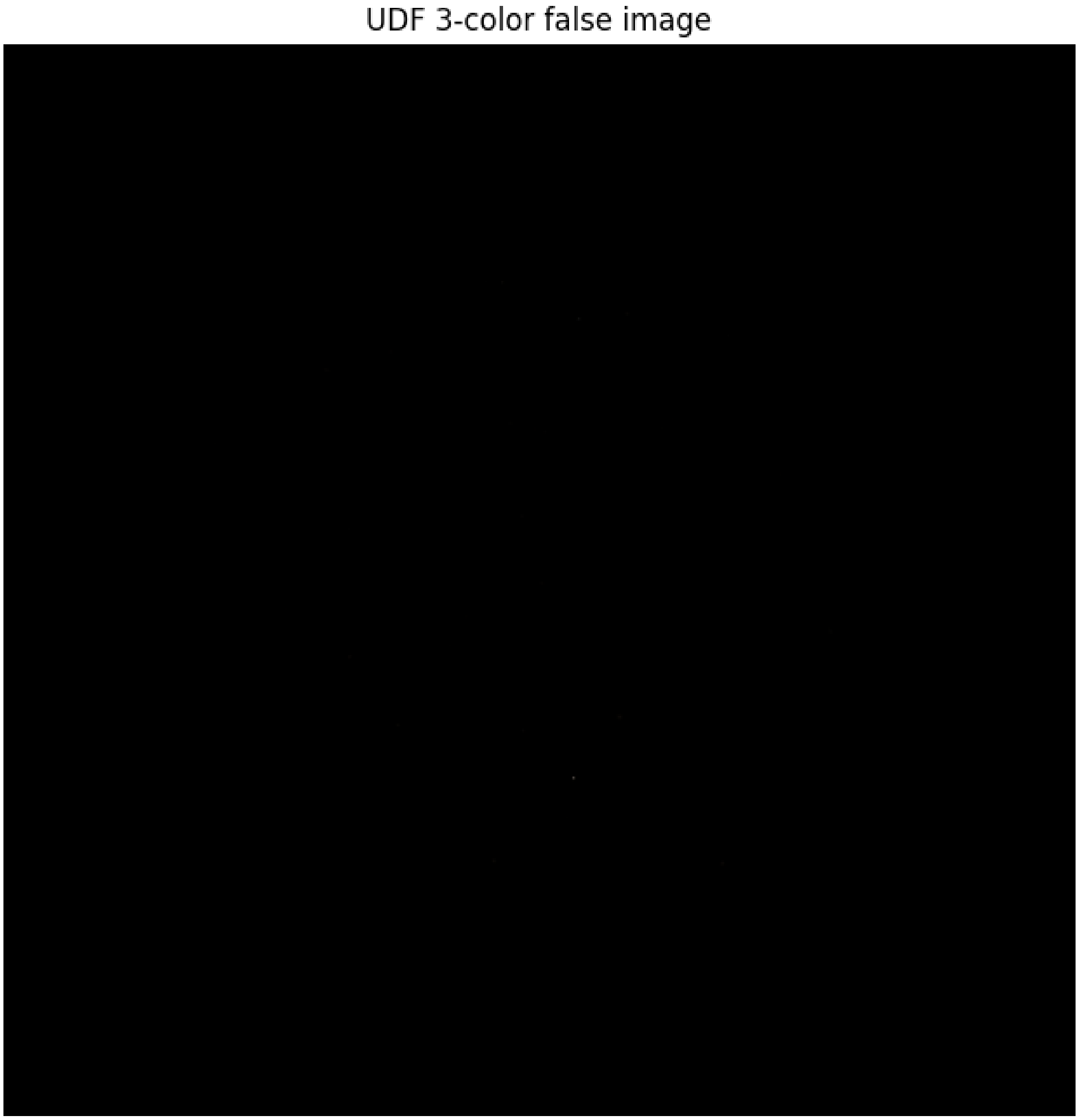
```
rgb_image = np.zeros((f160w_data.shape[0], f160w_data.shape[1], 3))
```

[66]:

```
rgb_image[:, :, 0] = f160w_data
rgb_image[:, :, 1] = f125w_data
rgb_image[:, :, 2] = f105w_data
```

[67]:

```
rgb_image = np.clip(rgb_image, 0, 1)
plt.figure(figsize=(8, 8))
plt.imshow(rgb_image, origin='lower')
plt.axis('off')
plt.title('UDF 3-color false image')
plt.show()
```



1 #not sure why my image is black

```
[ ]:
```